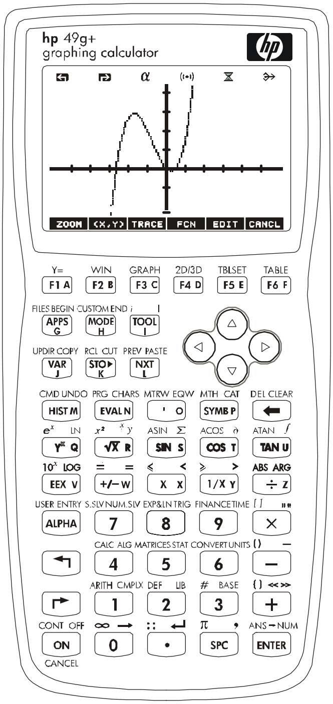
# hp 49g+

## graphing calculator

# user's guide



hp 49g+
graphing calculator
ZOOM (X,Y) TRACE FCN EDIT CANCL
Y= WIN GRAPH 2D/3D TBLSET TABLE
F1 A F2 B F3 C F4 D F5 E F6 F
FILES BEGIN CUSTOM END i I
APPS G MODE H TOOL I
UPDIR COPY RCL CUT PREV PASTE
VAR J STO K NXT L
CMD UNDO PRG CHARS MTRW EQW MTH CAT DEL CLEAR
HIST M EVAL N ' O SYMB P
ASIN ACOS ATAN
SIN S COS T TAN U
10^x LOG ABS ARG
EEX V +/− W X X 1/X Y ÷ Z
USER ENTRY S.SLV NUM.SLV EXP&LN TRIG FINANCE TIME
ALPHA 7 8 9 ×
CALC ALG MATRICES STAT CONVERT UNITS
4 5 6 −
ARITH CMPLX DEF LIB # BASE
1 2 3 +
CONT OFF ANS →NUM
ON 0 • SPC ENTER
CANCEL

invent

**NOTICE**

This manual and any examples contained herein are provided as-is and are subject to change without notice. Except to the extent prohibited by law, Hewlett-Packard Company makes no express or implied warranty of any kind with regard to this manual and specifically disclaims the implied warranties and conditions of merchantability and fitness for a particular purpose and Hewlett-Packard Company shall not be liable for any errors or for incidental or consequential damages in connection with the furnishing, performance or use of this manual and the examples herein.

# Preface

You have in your hands a compact symbolic and numerical computer that will facilitate calculation and mathematical analysis of problems in a variety of disciplines, from elementary mathematics to advanced engineering and science subjects. Although referred to as a calculator, because of its compact format resembling typical hand-held calculating devices, the hp 49 g+ should be thought of as a graphics/programmable hand-held computer.

The hp 49 g+ can be operated in two different calculating modes, the *Reverse Polish Notation* (RPN) mode and the *Algebraic* (ALG) mode (see page 1-11 in user's guide for additional details). The RPN mode was incorporated into calculators to make calculations more efficient. In this mode, the operands in an operation (e.g., '*2*' and '*3*' in the operation '*2+3*') are entered into the calculator screen, referred to as the *stack*, and then the operator (e.g., '*+*' in the operation '*2+3*') is entered to complete the operation. The ALG mode, on the other hand, mimics the way you type arithmetic expressions in paper. Thus, the operation '*2+3*', in ALG mode, will be entered in the calculator by pressing the keys '*2*', '*+*', and '*3*', in that order. To complete the operation we use the ENTER key. Examples of applications of the different functions and operations in this calculator are illustrated in this user's guide in both modes.

This guide contains examples that illustrate the use of the basic calculator functions and operations. The chapters in this user's guide are organized by subject in order of difficulty. Starting with the setting of calculator modes and display options, and continuing with real and complex number calculations, operations with lists, vectors, and matrices, detailed examples of graph applications, use of strings, basic programming, graphics programming, string manipulation, advanced calculus and multivariate calculus applications, advanced differential equations applications (including Laplace transform, and Fourier series and transforms), and probability and statistic applications.

For symbolic operations the calculator includes a powerful Computer Algebraic System (CAS) that lets you select different modes of operation, e.g., complex numbers vs. real numbers, or exact (symbolic) vs. approximate (numerical) mode. The display can be adjusted to provide textbook-type expressions, which can be useful when working with matrices, vectors, fractions, summations, derivatives, and integrals. The high-speed graphics of the calculator are very convenient for producing complex figures in very little time.

Thanks to the infrared port and the USB cable available with your calculator, you can connect your calculator with other calculators or computers. The high-speed connection through infrared or USB allows the fast and efficient exchange of programs and data with other calculators or computers. The calculator provides flash memory card ports to facilitate storage and exchange of data with other users.

The programming capabilities of the calculator allow you or other users to develop efficient applications for specific purposes. Whether it is advanced mathematical applications, specific problem solution, or data logging, the programming languages available in your calculator make it into a very versatile computing device.

We hope your calculator will become a faithful companion for your school and professional applications.

# Table of contents

## Appendices

# Chapter 1
# Getting started

This chapter is aimed at providing basic information in the operation of your calculator.  The exercises are aimed at familiarizing yourself with the basic operations and settings before actually performing a calculation.

## Basic Operations

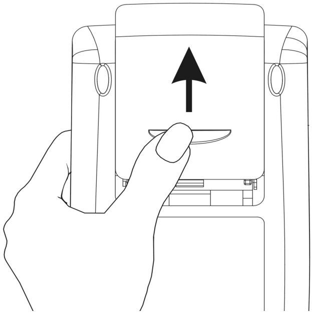
The following exercises are aimed at getting you acquainted with the hardware of your calculator.

### Batteries

The calculator uses 3 AAA batteries as main power and a CR2032 lithium battery for memory backup.
Before using the calculator, please install the batteries according to the following procedure.

**To install the main batteries**

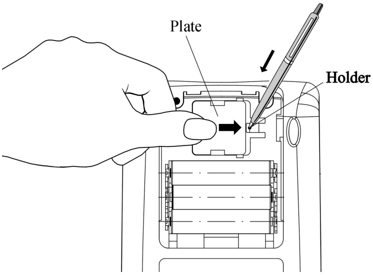
a. Slide up the battery compartment cover as illustrated.



b. Insert 3 new AAA batteries into the main compartment. Make sure each battery is inserted in the indicated direction.

**To install the backup battery**

a.  Press down the holder. Push the plate to the shown direction and lift it.



b. Insert a new CR2032 lithium battery. Make sure its positive (+) side is facing up.

c. Replace the plate and push it to the original place.

After installing the batteries, press [ON] to turn the power on.

**Warning:** When the low battery icon is displayed, you need to replace the batteries as soon as possible. However, avoid removing the backup battery and main batteries at the same time to avoid data lost.

## Turning the calculator on and off

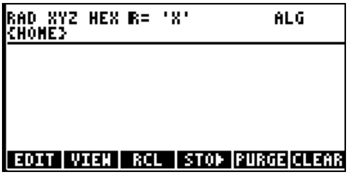
The ON key is located at the lower left corner of the keyboard. Press it once to turn your calculator on. To turn the calculator off, press the red right-shift key ↱ (first key in the second row from the bottom of the keyboard), followed by the ON key. Notice that the ON key has a red OFF label printed in the upper right corner as a reminder of the OFF command.

## Adjusting the display contrast

You can adjust the display contrast by holding the ON key while pressing the + or − keys. The ON (hold) + key combination produces a darker display. The ON (hold) − key combination produces a lighter display

## Contents of the calculator's display

Turn your calculator on once more. The display should look as indicated below.



```
RAD XYZ HEX R= 'X'        ALG
{HOME}
```

At the top of the display you will have two lines of information that describe the settings of the calculator. The first line shows the characters:

`RAD XYZ HEX R= 'X'`

For details on the meaning of these specifications see Chapter 2.
The second line shows the characters: `{ HOME }` indicating that the HOME directory is the current file directory in the calculator's memory. In Chapter 2 you will learn that you can save data in your calculator by storing them in files or variables. Variables can be organized into directories and sub-directories. Eventually, you may create a branching tree of file directories, similar to those in a computer hard drive. You can then navigate through the file directory tree to select any directory of interest. As you navigate through the file directory the second line of the display will change to reflect the proper file directory and sub-directory.

At the bottom of the display you will find a number of labels, namely,

EDIT VIEW RCL STO PURGE CLEAR

associated with the six *soft menu keys*, F1 through F6:

F1 F2 F3 F4 F5 F6

The six labels displayed in the lower part of the screen will change depending on which menu is displayed. But F1 will always be associated with the first displayed label, F2 with the second displayed label, and so on.

## Menus

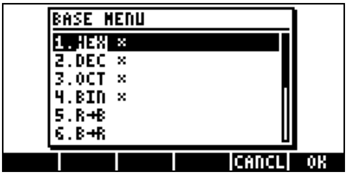
The six labels associated with the keys F1 through F6 form part of a menu of functions. Since the calculator has only six soft menu keys, it only display 6 labels at any point in time. However, a menu can have more than six entries. Each group of 6 entries is called a Menu page. The current menu, known as the TOOL menu (see below), has eight entries arranged in two pages. The next page, containing the next two entries of the menu is available by pressing the NXT (NeXT menu) key. This key is the third key from the left in the third row of keys in the keyboard. Press NXT once more to return to the main TOOL menu, or press the TOOL key (third key in second row of keys from the top of the keyboard).

The TOOL menu is described in detain in the next section. At this point we will illustrate some properties of menus that you will find useful while using your calculator.

**SOFT menus vs. CHOOSE boxes**

Menus, or SOFT menus, associate labels in the lower part of the screen with the six soft menu keys (F1 through F6). By pressing the appropriate soft menu key, the function shown in the associated label gets activated. For example, with the TOOL menu active, pressing the CLEAR key (F6) activates function CLEAR, which erases (clears up) the contents of the screen. To see this function in action, type a number, say 1 2 3 ENTER, and then press the F6 key.

SOFT menus are typically used to select from among a number of related functions. However, SOFT menus are not the only way to access collections of related functions in the calculator. The alternative way will be referred to as CHOOSE boxes. To see an example of a choose box, activate the TOOL menu (press TOOL), and then press the keystroke combination ↱ BASE (associated with the 3 key). This will provide the following CHOOSE box:



This CHOOSE box is labeled BASE MENU and provides a list of numbered functions, from 1. HEX x to 6. B→R. This display will constitute the first page of this CHOOSE box menu showing six menu functions. You can navigate through the menu by using the up and down arrow keys, ▲▼, located in the upper right side of the keyboard, right under the F5 and F6 soft menu keys. To activate any given function, first, highlight the function name by using the up and down arrow keys, ▲▼, or by pressing the number corresponding to the function in the CHOOSE box. After the function name is selected, press the OK soft menu key (F6). Thus, if you wanted to use function R→B (Real to Binary), you could press 6 F6.

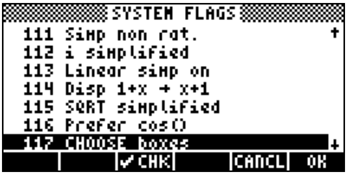
If you want to move to the top of the current menu page in a CHOOSE box, use ↰▲. To move to the bottom of the current page, use ↰▼. To move to the top of the entire menu, use ↱▲. To move to the bottom of the entire menu, use ↱▼.

### Selecting SOFT menus or CHOOSE boxes

You can select the format in which your menus will be displayed by changing a setting in the calculator system flags (A system flag is a calculator variable that controls a certain calculator operation or mode. For more information about flags, see Chapter 24). System flag 117 can be set to produce either SOFT menus or CHOOSE boxes. To access this flag use:

MODE FLAGS ▲↰ ▲▼

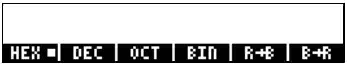
Your calculator will show the following screen, highlighting the line starting with the number 117:



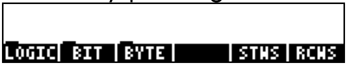
```
SYSTEM FLAGS
111 Simp non rat.
112 i simplified
113 Linear simp on
114 Disp 1+x → x+1
115 SQRT simplified
116 Prefer cos()
117 CHOOSE boxes
```

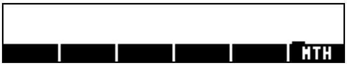
By default, the line will look as shown above. The highlighted line (117 CHOOSE boxes) indicates that CHOOSE boxes are the current menu display setting. If you prefer to use SOFT menu keys, press the CHK soft menu key (F3), followed by OK (F6). Press OK (F6) once more to return to normal calculator display.

If you now press ↱ BASE, instead of the CHOOSE box that you saw earlier, the display will now show six soft menu labels as the first page of the STACK menu:



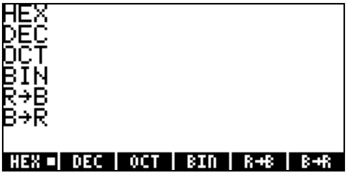
To navigate through the functions of this menu, press the NXT key to move to the next page, or ← PREV (associated with the NXT key) to move to the previous page. The following figures show the different pages of the BASE menu accessed by pressing the NXT key twice:

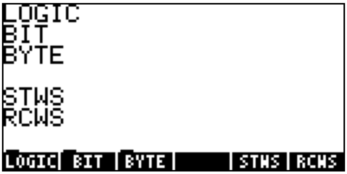




Pressing the NXT key once more will takes us back to the first menu page.

**Note:** With the SOFT menu setting for system flag 117, the keystroke combination →(hold) ▼, will show a list of the functions in the current soft menu. For example, for the two first pages in the BASE menu, you will get:


HEX
DEC
OCT
BIN
R→B
B→R


LOGIC
BIT
BYTE

STWS
RCWS

To revert to the CHOOSE boxes setting, use:

MODE FLAGS ▲ ← ▲ ▼ CHK OK OK.

**Notes:**

1. The TOOL menu, obtained by pressing TOOL, will always produce a SOFT menu.

2. Most of the examples in this User's Manual are shown using both SOFT menus and CHOOSE boxes. Programming applications (Chapters 21 and 22) use exclusively SOFT menus.

3. Additional information on SOFT menus vs. CHOOSE boxes is presented in Chapter 2 o f this guide.

## The TOOL menu

The soft menu keys for the menu currently displayed, known as the TOOL menu, are associated with operations related to manipulation of variables (see pages for more information on variables):

EDIT F1 EDIT the contents of a variable (see Chapter 2 and Appendix L for more information on editing)

VIEW F2 VIEW the contents of a variable
RCL F3 ReCaLl the contents of a variable
STO F4 STOre the contents of a variable
PURGE F5 PURGE a variable
CLEAR F6 CLEAR the display or stack

The calculator has only six soft menu keys, and can only display 6 labels at any point in time. However, a menu can have more than six entries. Each group of 6 entries is called a Menu page. The TOOL menu has eight entries arranged in two pages. The next page, containing the next two entries of the menu are available by pressing the NXT (NeXT menu) key. This key is the third key from the left in the third row of keys in the keyboard.

In this case, only the first two soft menu keys have commands associated with them. These commands are:

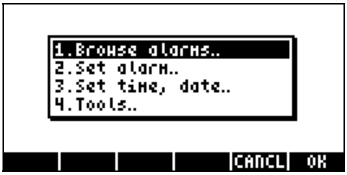
CASCM F1 CASCMD: CAS CoMmanD, used to launch a command from the CAS by selecting from a list
HELP F2 HELP facility describing the commands available

Pressing the NXT key will show the original TOOL menu. Another way to recover the TOOL menu is to press the TOOL key (third key from the left in the second row of keys from the top of the keyboard).

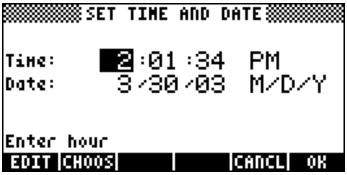
## Setting time and date

The calculator has an internal real time clock. This clock can be continuously displayed on the screen and be used for alarms as well as running scheduled tasks. This section will show not only how to set time and date, but also the basics of using CHOOSE boxes and entering data in a dialog box. Dialog boxes on your calculator are similar to a computer dialog box.

To set time and date we use the TIME choose box available as an alternative function for the 9 key. By combining the red right-shift button, ↱, with the 9 key the TIME choose box is activated. This operation can also be represented as ↱ TIME . The TIME choose box is shown in the figure below:



As indicated above, the TIME menu provides four different options, numbered 1 through 4. Of interest to us as this point is option *3. Set time, date...* Using the down arrow key, ▼, highlight this option and press the OK F6 soft menu key. The following *input form* (see Appendix 1-A) for adjusting time and date is shown:

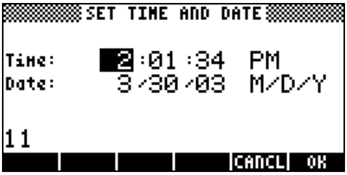

SET TIME AND DATE

Time: 2:01:34 PM
Date: 3/30/03 M/D/Y

Enter hour

### Setting the time of the day

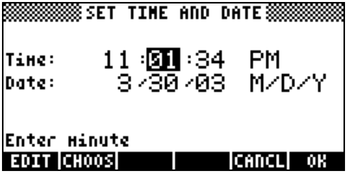
Using the number keys, 1 2 3 4 5 6 7 8 9 0, start by adjusting the hour of the day. Suppose that we change the hour to 11, by pressing 1 1 as the hour field in the SET TIME AND DATE input form is highlighted. This results in the number *11* being entered in the lower line of the input form:


SET TIME AND DATE

Time: 2:01:34 PM
Date: 3/30/03 M/D/Y

11

Press the OK F6 soft menu key to effect the change. The value of *11* is now shown in the hour field, and the minute field is automatically highlighted:


SET TIME AND DATE

Time: 11:01:34 PM
Date: 3/30/03 M/D/Y

Enter minute

Let's change the minute field to 25, by pressing: 2 5 OK . The seconds field is now highlighted. Suppose that you want to change the seconds field to 45, use: 4 5 OK

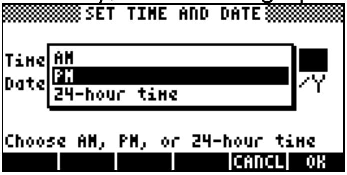
The time format field is now highlighted. To change this field from its current setting you can either press the +/- key (the second key from the left in the fifth row of keys from the bottom of the keyboard), or press the CHOOS soft menu key ( F2 ).

- If using the +/- key, the setting in the time format field will change to either of the following options:

  - o AM : indicates that displayed time is AM time
  - o PM : indicates that displayed time is PM time
  - o 24-hr : indicates that that the time displayed uses a 24 hour format where18:00, for example, represents 6pm

  The last selected option will become the set option for the time format by using this procedure.

- If using the CHOOS soft menu key, the following options are available.



SET TIME AND DATE

Time AM
Date PM
24-hour time

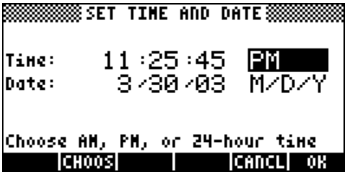
Choose AM, PM, or 24-hour time

CANCL OK

Use the up and down arrow keys, ▲ ▼, to select among these three options (AM, PM, 24-hour time). Press the OK F6 soft menu key to make the selection.

### Setting the date

After setting the time format option, the SET TIME AND DATE input form will look as follows:

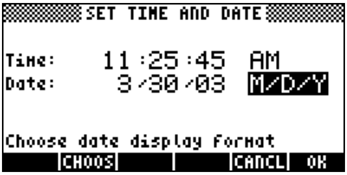


SET TIME AND DATE

Time: 11:25:45 PM
Date: 3/30/03 M/D/Y

Choose AM, PM, or 24-hour time

To set the date, first set the date format. The default format is M/D/Y (month/day/year). To modify this format, press the down arrow key. This will highlight the date format as shown below:

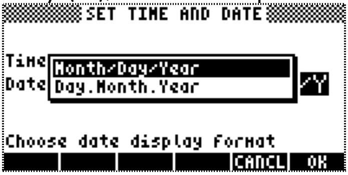


SET TIME AND DATE

Time: 11:25:45 AM
Date: 3/30/03 M/D/Y

Choose date display format

Use the CHOOS soft menu key ( B), to see the options for the date format:



SET TIME AND DATE

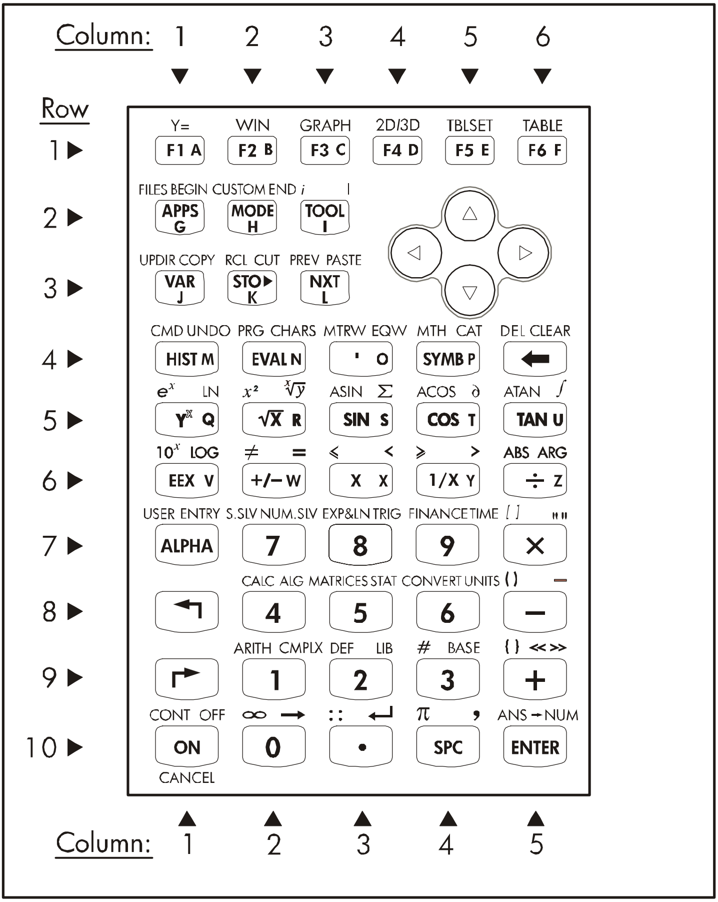
Month/Day/Year
Day.Month.Year

Choose date display format

Highlight your choice by using the up and down arrow keys, ▲ ▼, and press the OK F6 soft menu key to make the selection.

# Introducing the calculator's keyboard

The figure below shows a diagram of the calculator's keyboard with the numbering of its rows and columns.

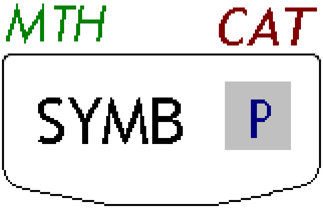

Column: 1 2 3 4 5 6
Row 1 2 3 4 5 6 7 8 9 10
Y= WIN GRAPH 2D/3D TBLSET TABLE
F1 A F2 B F3 C F4 D F5 E F6 F
FILES BEGIN CUSTOM END i I
APPS G MODE H TOOL I
UPDIR COPY RCL CUT PREV PASTE
VAR J STO▶ K NXT L
CMD UNDO PRG CHARS MTRW EQW MTH CAT DEL CLEAR
HIST M EVAL N ' O SYMB P
e^x LN x² ˣ√y ASIN Σ ACOS ∂ ATAN ∫
Yˣ Q √X R SIN S COS T TAN U
10ˣ LOG ≠ = ≤ < ≥ > ABS ARG
EEX V +/− W X X 1/X Y ÷ Z
USER ENTRY S.SLV NUM.SLV EXP&LN TRIG FINANCE TIME [ ] " "
ALPHA 7 8 9 ×
CALC ALG MATRICES STAT CONVERT UNITS ( ) _
4 5 6 −
ARITH CMPLX DEF LIB # BASE { } « »
1 2 3 +
CONT OFF ∞ → : : ↵ π , ANS →NUM
ON 0 • SPC ENTER
CANCEL
Column: 1 2 3 4 5

The figure shows 10 rows of keys combined with 3, 5, or 6 columns. Row 1 has 6 keys, rows 2 and 3 have 3 keys each, and rows 4 through 10 have 5 keys each. There are 4 arrow keys located on the right-hand side of the keyboard in the space occupied by rows 2 and 3.

Each key has three, four, or five functions. The main key function correspond to the most prominent label in the key. Also, the green left-shift key, *key (8,1)*, the red right-shift key, *key (9,1)*, and the blue ALPHA key, *key (7,1)*, can be combined with some of the other keys to activate the alternative functions shown in the keyboard. For example, the SYMB key, *key(4,4)*, has the following six functions associated with it:

SYMB — Main function, to activate the SYMBolic menu
← MTH — Left-shift function, to activate the MTH (Math) menu
→ CAT — Right-shift function, to activate the CATalog function
ALPHA P — ALPHA function, to enter the upper-case letter P
ALPHA ← P — ALPHA-Left-Shift function, to enter the lower-case letter p
ALPHA → P — ALPHA-Right-Shift function, to enter the symbol P

Of the six functions associated with the key only the first four are shown in the keyboard itself. This is the way that the key looks in the keyboard:



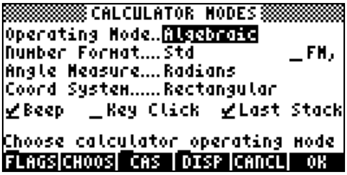
Notice that the color and the position of the labels in the key, namely, **SYMB**, *MTH*, *CAT* and **P**, indicate which is the main function (**SYMB**), and which of the other three functions is associated with the left-shift ← (*MTH*), right-shift → (*CAT* ) , and ALPHA (**P**) keys.

For detailed information on the calculator keyboard operation referee to Appendix B .

## Selecting calculator modes

This section assumes that you are now at least partially familiar with the use of choose and dialog boxes (if you are not, please refer to Chapter 2).

Press the MODE button (second key from the left on the second row of keys from the top) to show the following *CALCULATOR MODES* input form:



```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Std          _FM,
Angle Measure....Radians
Coord System......Rectangular
✓Beep   _Key Click   ✓Last Stack

Choose calculator operating mode
FLAGS|CHOOS| CAS |DISP |CANCL| OK
```

Press the OK (F6) soft menu key to return to normal display. Examples of selecting different calculator modes are shown next.

## Operating Mode

The calculator offers two operating modes: the *Algebraic* mode, and the *Reverse Polish Notation* (*RPN*) mode. The default mode is the Algebraic mode (as indicated in the figure above), however, users of earlier HP calculators may be more familiar with the RPN mode.

To select an operating mode, first open the CALCULATOR MODES input form by pressing the (MODE) button. The *Operating Mode* field will be highlighted. Select the *Algebraic* or *RPN* operating mode by either using the (+/-) key (second from left in the fifth row from the keyboard bottom), or pressing the CHOOS soft menu key ( (F2) ). If using the latter approach, use up and down arrow keys, (▲) (▼), to select the mode, and press the OK soft menu key to complete the operation.

To illustrate the difference between these two operating modes we will calculate the following expression in both modes:

$$\sqrt{\frac{3\cdot\left(5-\frac{1}{3\cdot 3}\right)}{23^3}+e^{2.5}}$$

To enter this expression in the calculator we will first use the *equation writer*, (↱) EQW . Please identify the following keys in the keyboard, besides the numeric keypad keys:

(↰)(↱)(•)(SPC)(×)(+)(−)(÷)(√x)
(yˣ) eˣ () (↱) EQW (◀)(▶)(▼)(▲)(ENTER)

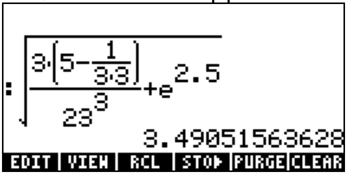
The equation writer is a display mode in which you can build mathematical expressions using explicit mathematical notation including fractions, derivatives, integrals, roots, etc. To use the equation writer for writing the expression shown above, use the following keystrokes:

↱ EQW √x 3 × ↰ () 5 −
1 ÷ 3 × 3
▲ ▲ ▲ ▲ ▲ ▲ ▲
÷ 2 3 $Y^x$ 3 ▶ ▶ + ↰ $e^x$ 2 • 5 ENTER

After pressing ENTER the calculator displays the expression:

```
√(3*(5-1/(3*3))/23^3+EXP(2.5))
```

Pressing ENTER again will provide the following value. Accept Approx. mode on, if asked, by pressing OK. [**Note**: The integer values used above, e.g., 3, 5, 1, represent exact values. The EXP(2.5), however, cannot be expressed as an exact value, therefore, a switch to Approx mode is required]:



$$:\sqrt{\frac{3\cdot\left(5-\frac{1}{3\cdot3}\right)}{23^3}+e^{2.5}}$$

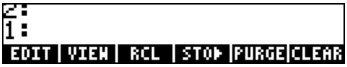
3.49051563628

You could also type the expression directly into the display without using the equation writer, as follows:

√x ↰ () 3 • × ↰ () 5 • −
1 • ÷ ↰ () 3 • × 3 • ▶ ▶
÷ 2 3 • $Y^x$ 3 + ↰ $e^x$ 2 • 5 ENTER

to obtain the same result.

Change the operating mode to RPN by first pressing the MODE button. Select the *RPN* operating mode by either using the +/- key, or pressing the CHOOS soft menu key. Press the OK F6 soft menu key to complete the operation. The display, for the RPN mode looks as follows:



2:
1:

Notice that the display shows several levels of output labeled, from bottom to top, as 1, 2, 3, etc. This is referred to as the *stack* of the calculator. The different levels are referred to as the *stack levels*, i.e., stack level 1, stack level 2, etc.

Basically, what RPN means is that, instead of writing an operation such as 3 + 2, in the calculator by using 3 + 2 ENTER, we write first the operands, in the proper order, and then the operator, i.e., 3 ENTER 2 ENTER +. As you enter the operands, they occupy different stack levels. Entering 3 ENTER puts the number 3 in stack level 1. Next, entering 2 ENTER pushes the 3 upwards to occupy stack level 2. Finally, by pressing +, we are telling the calculator to apply the operator, or program, + to the objects occupying levels 1 and 2. The result, 5, is then placed in level 1. A simpler way to calculate this operation is by using: 3 ENTER 2 +.

Let's try some other simple operations before trying the more complicated expression used earlier for the algebraic operating mode:

| | |
|---|---|
| 123/32 | 1 2 3 ENTER 3 2 ÷ |
| $4^2$ | 4 ENTER 2 Yˣ |
| $^3\sqrt{27}$ | 2 7 ENTER 3 ↱ ˣ√y |

Notice the position of the y and the x in the last two operations. The base in the exponential operation is y (stack level 2) while the exponent is x (stack level 1) before the key Yˣ is pressed. Similarly, in the cubic root operation, y (stack level 2) is the quantity under the root sign, and x (stack level 1) is the root.

Try the following exercise involving 3 factors: (5 + 3) × 2

| | |
|---|---|
| 5 ENTER 3 + | Calculates (5 +3) first. |
| 2 X | Completes the calculation. |

Let's try now the expression proposed earlier:

$$\sqrt{\frac{3\cdot\left(5-\frac{1}{3\cdot 3}\right)}{23^3}+e^{2.5}}$$

| | |
|---|---|
| 3 • ENTER | Enter 3 in level 1 |
| 5 • ENTER | Enter 5 in level 1, 3 moves to y |
| 3 • ENTER | Enter 3 in level 1, 5 moves to level 2, 3 to level 3 |
| 3 • × | Place 3 and multiply, 9 appears in level 1 |
| 1/x | 1/(3×3), last value in lev. 1; 5 in level 2; 3 in level 3 |
| − | 5 - 1/(3×3) , occupies level 1 now; 3 in level 2 |
| × | 3× (5 - 1/(3×3)), occupies level 1 now. |
| 2 3 • ENTER | Enter 23 in level 1, 14.66666 moves to level 2. |
| 3 • Y^x | Enter 3, calculate $23^3$ into level 1. 14.666 in lev. 2. |
| ÷ | (3× (5-1/(3×3)))/$23^3$ into level 1 |
| 2 • 5 | Enter 2.5 level 1 |
| ↰ e^x | $e^{2.5}$, goes into level 1, level 2 shows previous value. |
| + | (3× (5 - 1/(3×3)))/$23^3$ + $e^{2.5}$ = 12.18369, into lev. 1. |
| √x | √((3× (5 - 1/(3×3)))/$23^3$ + $e^{2.5}$) = 3.4905156, into 1. |

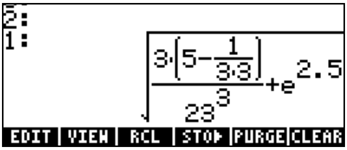
Although RPN requires a little bit more thought than the algebraic (ALG) mode, there are multiple advantages in using RPN. For example, in RPN mode you can see the equation unfolding step by step. This is extremely useful to detect a possible input error. Also, as you become more efficient in this mode and learn more of the tricks, you will be able to calculate expression faster and will much less keystrokes. Consider, for example the calculation of (4×6 - 5)/(1+4×6 - 5). In RPN mode you can write:

4 ENTER 6 × 5 − ENTER 1 + ÷

obviously, even In RPN mode, you can enter an expression in the same order as the algebraic mode by using the Equation writer. For example,

↱ EQW √x 3 • × ↰ () 5 • − 1 ÷ 3 • × 3 •
▲ ▲ ▲ ▲ ▲ ▲ ▲
÷ 2 3 • Y^x 3 ▶ ▶ + ↰ e^x 2 • 5 ENTER

The resulting expression is shown in stack level 1 as follows:



```
2:
1:
```

$$\sqrt{\frac{3\cdot\left(5-\frac{1}{3\cdot3}\right)}{23^{3}}+e^{2.5}}$$

Notice how the expression is placed in stack level 1 after pressing ENTER. Pressing the EVAL key at this point will evaluate the numerical value of that expression Note: In RPN mode, pressing ENTER when there is no command line will execute the DUP function which copies the contents of stack level 1 of the stack onto level 2 (and pushes all the other stack levels one level up). This is extremely useful as showed in the previous example.

To select between the ALG vs. RPN operating mode, you can also set/clear system flag 95 through the following keystroke sequence:

MODE FLAGS ▲↰▲↰▲↰ ▲ CHK OK OK

Alternatively, you can use one of the following shortcuts:

- In ALG mode,
  CF(-95) selects RPN mode

- In RPN mode,
  95 +/- ENTER SF selects ALG mode

For more information on calculator's system flags see Chapter 2.

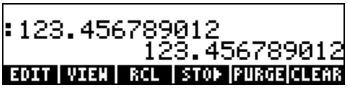
## Number Format and decimal dot or comma

Changing the number format allows you to customize the way real numbers are displayed by the calculator. You will find this feature extremely useful in operations with powers of tens or to limit the number of decimals in a result.

To select a number format, first open the CALCULATOR MODES input form by pressing the MODE button. Then, use the down arrow key, ▼, to select the option *Number format*. The default value is *Std*, or *St*and*d*ard format. In the standard format, the calculator will show floating-point numbers with the maximum precision allowed by the calculator (12 significant digits). To learn more about reals, see Chapter 2. To illustrate this and other number formats try the following exercises:

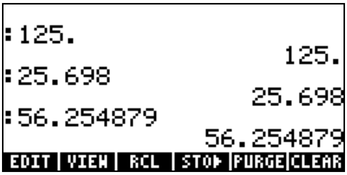
- **Standard format**:
  This mode is the most used mode as it shows numbers in the most familiar notation.
  Press the OK soft menu key, with the *Number format* set to *Std*, to return to the calculator display. Enter the number 123.4567890123456. Notice that this number has 16 significant figures. Press the ENTER key.

The number is rounded to the maximum 12 significant figures, and is displayed as follows:



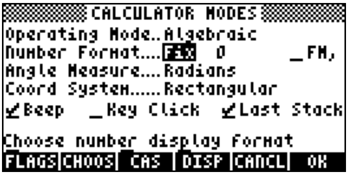
```
:123.456789012
                 123.456789012
```

In the standard format of decimal display, integer numbers are shown with no decimal zeros whatsoever. Numbers with different decimal figures will be adjusted in the display so that only those decimal figures that are necessary will be shown. More examples of numbers in standard format are shown next:



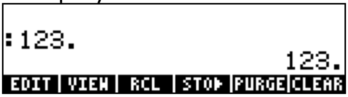
```
:125.
                          125.
:25.698
                        25.698
:56.254879
                     56.254879
```

- **Fixed format with no decimals**: Press the MODE button. Next, use the down arrow key, ▼, to select the option *Number format.* Press the CHOOS soft menu key ( F2 ), and select the option *Fixed* with the arrow down key ▼ .



```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Fix 0      _FM,
Angle Measure....Radians
Coord System......Rectangular
✓Beep   _Key Click   ✓Last Stack

Choose number display format
```

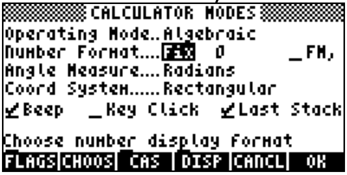
Notice that the Number Format mode is set to *Fix* followed by a zero (*0*). This number indicates the number of decimals to be shown after the decimal point in the calculator's display. Press the OK soft menu key to return to the calculator display. The number now is shown as:



```
:123.
                   123.
```

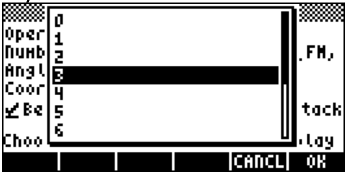
This setting will force all results to be rounded to the closest integer (0 digit displayed after the comma). However, the number is still stored by the calculator with its full 12 significant digit precision. As we change the number of decimals to be displayed, you will see the additional digits being shown again.

- **Fixed format with decimals**:
  This mode is mainly used when working with limited precision. For example, if you are doing financial calculation, using a FIX 2 mode is convenient as it can easily represent monetary units to a 1/100 precision. Press the MODE button. Next, use the down arrow key, ▼, to select the option *Number format*. Press the CHOOS soft menu key ( F2 ), and select the option *Fixed* with the arrow down key ▼.



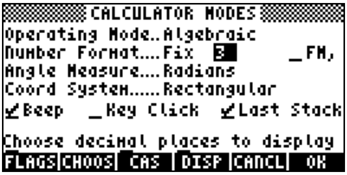
```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Fix  0      _FM,
Angle Measure....Radians
Coord System......Rectangular
✓Beep   _Key Click  ✓Last Stack
Choose number display format
FLAGS|CHOOS| CAS |DISP |CANCL| OK
```

Press the right arrow key, ▶, to highlight the zero in front of the option *Fix*. Press the CHOOS soft menu key and, using the up and down arrow keys, ▲▼, select, say, 3 decimals.



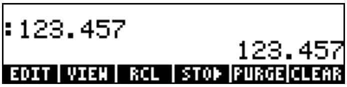
```
Oper 0
Numb 1       .FM,
Angl 2
Coor 3
✓Be  4       tack
Choo 5       .lay
     6
|     |     |     |     |CANCL| OK
```

Press the OK soft menu key to complete the selection:



```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Fix  3      _FM,
Angle Measure....Radians
Coord System......Rectangular
✓Beep   _Key Click  ✓Last Stack
Choose decimal places to display
FLAGS|CHOOS| CAS |DISP |CANCL| OK
```

Press the OK soft menu key return to the calculator display. The number now is shown as:



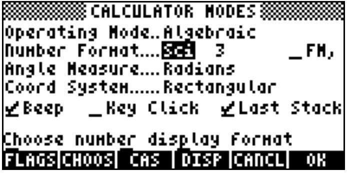
```
:123.457
                     123.457
EDIT | VIEW | RCL | STO▶|PURGE|CLEAR
```

Notice how the number is rounded, not truncated. Thus, the number 123.4567890123456, for this setting, is displayed as 123.457, and not as 123.456 because the digit after 6 is > 5):

- **Scientific format**

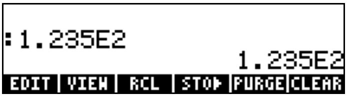
  The scientific format is mainly used when solving problems in the physical sciences where numbers are usually represented as a number with limited precision multiplied by a power of ten.

  To set this format, start by pressing the MODE button. Next, use the down arrow key, ▼, to select the option *Number format*. Press the CHOOSE soft menu key ( F2 ), and select the option *Scientific* with the arrow down key ▼. Keep the number 3 in front of the *Sci*. (This number can be changed in the same fashion that we changed the *Fixed* number of decimals in the example above).



```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Sci  3      _FM,
Angle Measure....Radians
Coord System......Rectangular
✓Beep  _Key Click  ✓Last Stack

Choose number display format
```

  Press the OK soft menu key return to the calculator display. The number now is shown as:



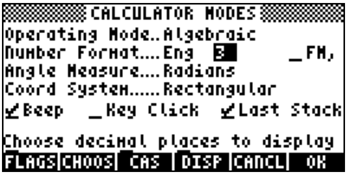
```
:1.235E2
                1.235E2
```

  This result, 1.23E2, is the calculator's version of powers-of-ten notation, i.e., $1.235 \times 10^{2}$. In this, so-called, scientific notation, the number *3* in front of the *Sci* number format (shown earlier) represents the number of significant figures after the decimal point. Scientific notation always includes one integer figure as shown above. For this case, therefore, the number of significant figures is four.

- **Engineering format**

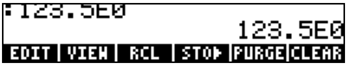
  The engineering format is very similar to the scientific format, except that the powers of ten are multiples of three.

  To set this format, start by pressing the MODE button. Next, use the down arrow key, ▼, to select the option *Number format*. Press the CHOOSE soft menu key ( F2 ), and select the option *Engineering* with the arrow down key ▼. Keep the number 3 in front of the *Eng*. (This number can be changed in the same fashion that we changed the *Fixed* number of decimals in an earlier example).



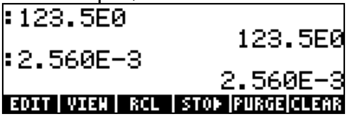
```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Eng 3      _FM,
Angle Measure....Radians
Coord System......Rectangular
✓Beep  _Key Click  ✓Last Stack

Choose decimal places to display
FLAGS CHOOS CAS DISP CANCL OK
```

Press the OK soft menu key return to the calculator display. The number now is shown as:



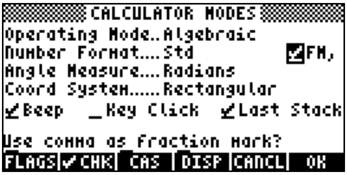
```
:123.5E0
                 123.5E0
EDIT VIEW RCL STO▶ PURGE CLEAR
```

Because this number has three figures in the integer part, it is shown with four significative figures and a zero power of ten, while using the Engineering format. For example, the number 0.00256, will be shown as:



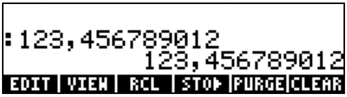
```
:123.5E0
                 123.5E0
:2.560E-3
                2.560E-3
EDIT VIEW RCL STO▶ PURGE CLEAR
```

- **Decimal comma vs. decimal point**
  Decimal points in floating-point numbers can be replaced by commas, if the user is more familiar with such notation. To replace decimal points for commas, change the *FM* option in the CALCULATOR MODES input form to commas, as follows (Notice that we have changed the *Number Format* to *Std*):
- Press the MODE button. Next, use the down arrow key, ▼, once, and the right arrow key, ▶, twice, to highlight the option *__FM,*. To select commas, press the CHECK soft menu key (i.e., the F2 key). The input form will look as follows:



```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Std        ☑FM,
Angle Measure....Radians
Coord System......Rectangular
✓Beep  _Key Click  ✓Last Stack

Use comma as fraction mark?
FLAGS ✓CHK CAS DISP CANCL OK
```

- Press the OK soft menu key return to the calculator display. The number 123.456789012, entered earlier, now is shown as:



```
:123,456789012
          123,456789012
EDIT VIEW RCL STO▶ PURGE CLEAR
```

## Angle Measure

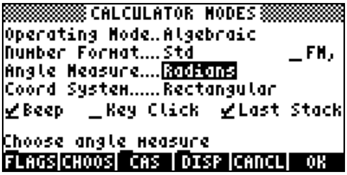
Trigonometric functions, for example, require arguments representing plane angles. The calculator provides three different *Angle Measure* modes for working with angles, namely:

- *Degrees*: There are *360* degrees (*360°*) in a complete circumference, or *90* degrees (*90°*) in a right angle. This representation is mainly used when doing basic geometry, mechanical or structural engineering, and surveying.
- *Radians*: There are $2\pi$ radians ($2\pi\ ^r$) in a complete circumference, or $\pi/2$ radians ($\pi/2\ ^r$) in a right angle. This notation is mainly used when solving mathematics and physics problems. This is the default mode of the calculator.
- *Grades*: There are *400* grades (*400* $^g$) in a complete circumference, or *100* grades (*100* $^g$) in a right angle. This notation is similar to the degree mode, and was introduced in order to "simplify" the degree notation but is seldom used now.

The angle measure affects the trig functions like SIN, COS, TAN and associated functions.

To change the angle measure mode, use the following procedure:

- Press the MODE button. Next, use the down arrow key, ▼, twice. Select the *Angle Measure* mode by either using the +/- key (second from left in the fifth row from the keyboard bottom), or pressing the CHOOSE soft menu key ( F2 ). If using the latter approach, use up and down arrow keys, ▲ ▼, to select the preferred mode, and press the OK F6 soft menu key to complete the operation. For example, in the following screen, the Radians mode is selected:



```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Std        _FM,
Angle Measure....Radians
Coord System......Rectangular
✓Beep   _Key Click  ✓Last Stack

Choose angle measure
FLAGS|CHOOS| CAS |DISP |CANCL| OK
```

## Coordinate System

The coordinate system selection affects the way vectors and complex numbers are displayed and entered. To learn more about complex numbers and vectors, see Chapters 4 and 9, respectively.

Two- and three-dimensional vector components and complex numbers can be represented in any of 3 coordinate systems: The Cartesian (2 dimensional) or Rectangular (3 dimensional), Cylindrical (3 dimensional) or Polar (2 dimensional), and Spherical (only 3 dimensional). In a Cartesian or Rectangular coordinate system a point P will have three linear coordinates (x,y,z) measured from the origin along each of three mutually perpendicular axes (in 2 d mode, z is assumed to be 0). In a Cylindrical or Polar coordinate system the coordinates of a point are given by $(r,\theta,z)$, where r is a radial distance measured from the origin on the xy plane, $\theta$ is the angle that the radial distance r forms with the positive x axis -- measured as positive in a counterclockwise direction --, and z is the same as the z coordinate in a Cartesian system (in 2 d mode, z is assumed to be 0). The Rectangular and Polar systems are related by the following quantities:

$$x = r \cdot \cos(\theta) \qquad\qquad r = \sqrt{x^2 + y^2}$$

$$y = r \cdot \sin(\theta) \qquad\qquad \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

$$z = z$$

In a Spherical coordinate system the coordinates of a point are given by $(\rho,\theta,\phi)$ where $\rho$ is a radial distance measured from the origin of a Cartesian system, $\theta$ is an angle representing the angle formed by the projection of the linear distance $\rho$ onto the xy axis (same as $\theta$ in Polar coordinates), and $\phi$ is the angle from the positive $z$ axis to the radial distance $\rho$. The Rectangular and Spherical coordinate systems are related by the following quantities:

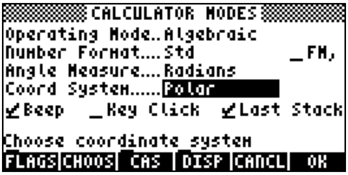
$$x = \rho \cdot \sin(\phi) \cdot \cos(\theta) \qquad\qquad \rho = \sqrt{x^2 + y^2 + z^2}$$

$$y = \rho \cdot \sin(\phi) \cdot \cos(\theta) \qquad\qquad \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

$$z = \rho \cdot \cos(\phi) \qquad\qquad \phi = \tan^{-1}\left(\frac{\sqrt{x^2 + y^2}}{z}\right)$$

To change the coordinate system in your calculator, follow these steps:

- Press the MODE button. Next, use the down arrow key, ▼, three times. Select the *Angle Measure* mode by either using the +/- key (second from left in the fifth row from the keyboard bottom), or pressing the CHOOS soft menu key ( F2 ). If using the latter approach, use up and down arrow keys, ▲ ▼, to select the preferred mode, and press the OK F6 soft menu key to complete the operation. For example, in the following screen, the Polar coordinate mode is selected:



```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Std        _FM,
Angle Measure....Radians
Coord System......Polar
✓Beep   _Key Click   ✓Last Stack

Choose coordinate system
FLAGS CHOOS CAS DISP CANCL OK
```

## Beep, Key Click, and Last Stack

The last line of the CALCULATOR MODES input form include the options:

*_Beep   _Key Click   _Last Stack*

By choosing the check mark next to each of these options, the corresponding option is activated. These options are described next:

*_Beep* : When selected, , the calculator beeper is active. This operation mainly applies to error messages, but also some user functions like BEEP.
*_Key Click* : When selected, each keystroke produces a "click" sound.
*_Last Stack*: Keeps the contents of the last stack entry for use with the functions UNDO and ANS (see Chapter 2).

The *_Beep* option can be useful to advise the user about errors. You may want to deselect this option if using your calculator in a classroom or library.
The *_Key Click* option can be useful as an audible way to check that each keystroke was entered as intended.
The *_Last Stack* option is very useful to recover the last operation in case we need it for a new calculation.

To select, or deselect, any of these three options, first press the MODE button. Next,

- Use the down arrow key, ˜, four times to select the *_Last Stack* option. Use the CHECK soft menu key (i.e., the F2 key) to change the selection.
- Press the left arrow key ◀ to select the *_Key Click* option. Use the CHECK soft menu key (i.e., the F2 key) to change the selection.
- Press the left arrow key ◀ to select the *_Beep* option. Use the CHECK soft menu key (i.e., the F2 key) to change the selection.
  Press the OK F6 soft menu key to complete the operation.

## Selecting CAS settings

CAS stands for *C*omputer *A*lgebraic *S*ystem. This is the mathematical core of the calculator where the symbolic mathematical operations and functions are programmed and performed. The CAS offers a number of settings can be adjusted according to the type of operation of interest. These are:

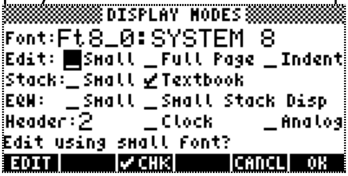
- The default independent variable
- Numeric vs. symbolic mode
- Approximate vs. Exact mode
- Verbose vs. Non-verbose mode
- Step-by-step mode for operations
- Increasing power format for polynomials
- Rigorous mode
- Simplification of non-rational expressions

Details on the selection of CAS settings are presented in Appendix C.

## Selecting Display modes

The calculator display can be customized to your preference by selecting different display modes. To see the optional display settings use the following:

- First, press the MODE button to activate the CALCULATOR MODES input form. Within the CALCULATOR MODES input form, press the DISP soft menu key (F4) to display the DISPLAY MODES input form.



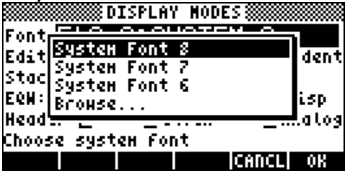
DISPLAY MODES
Font:Ft8_0:SYSTEM 8
Edit: ■Small _Full Page _Indent
Stack:_Small ✓Textbook
EQW: _Small _Small Stack Disp
Header:2 _Clock _Analog
Edit using small font?

- To navigate through the many options in the DISPLAY MODES input form, use the arrow keys: ◀ ▶ ▼ ▲ .
- To select or deselect any of the settings shown above, that require a check mark, select the underline before the option of interest, and toggle the CHK soft menu key until the right setting is achieved. When an option is selected, a check mark will be shown in the underline (e.g., the *Textbook* option in the *Stack:* line above). Unselected options will show no check mark in the underline preceding the option of interest (e.g., the *_Small, _Full page,* and *_Indent* options in the *Edit:* line above).
- To select the Font for the display, highlight the field in front of the *Font:* option in the DISPLAY MODES input form, and use the CHOOSE soft menu key (F2 ).
- After having selected and unselected all the options that you want in the DISPLAY MODES input form, press the OK soft menu key. This will take you back to the CALCULATOR MODES input form. To return to normal calculator display at this point, press the OK soft menu key once more.

## Selecting the display font

Changing the font display allows you to have the calculator look and feel changed to your own liking. By using a 6-pixel font, for example, you can display up to 9 stack levels! Follow these instructions to select your display font:

First, press the MODE button to activate the CALCULATOR MODES input form. Within the CALCULATOR MODES input form, press the DISP soft menu key (F4 ) to display the DISPLAY MODES input form. The *Font:* field is highlighted, and the option *Ft8_0:system 8* is selected. This is the default value of the display font. Pressing the CHOOSE soft menu key (F2 ), will provide a list of available system fonts, as shown below:



DISPLAY MODES
Font
Edit
Stac
EQW:
Head
System Font 8
System Font 7
System Font 6
Browse...
Choose system font

The options available are three standard *System Fonts* (sizes *8, 7,* and *6*) and a *Browse..* option. The latter will let you browse the calculator memory for additional fonts that you may have created (see Chapter 23) or downloaded into the calculator.

Practice changing the display fonts to sizes *7* and *6*. Press the OK soft menu key to effect the selection. When done with a font selection, press the OK soft menu key to go back to the CALCULATOR MODES input form. To return to normal calculator display at this point, press the OK soft menu key once more and see how the stack display change to accommodate the different font.

## Selecting properties of the line editor

First, press the MODE button to activate the CALCULATOR MODES input form. Within the CALCULATOR MODES input form, press the DISP soft menu key (F4) to display the DISPLAY MODES input form. Press the down arrow key, ▼, once, to get to the *Edit* line. This line shows three properties that can be modified. When these properties are selected (checked) the following effects are activated:

*_Small* Changes font size to small
*_Full page* Allows to place the cursor after the end of the line
*_Indent* Auto intend cursor when entering a carriage return

Detailed instructions on the use of the line editor are presented in Chapter 2 in this guide.

## Selecting properties of the Stack

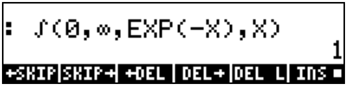
First, press the MODE button to activate the CALCULATOR MODES input form. Within the CALCULATOR MODES input form, press the DISP soft menu key (F4) to display the DISPLAY MODES input form. Press the down arrow key, ▼, twice, to get to the *Stack* line. This line shows two properties that can be modified. When these properties are selected (checked) the following effects are activated:

*_Small* Changes font size to small. This maximized the amount of information displayed on the screen. Note, this selection overrides the font selection for the stack display.
*_Textbook* Display mathematical expressions in graphical mathematical notation

To illustrate these settings, either in algebraic or RPN mode, use the equation writer to type the following definite integral:

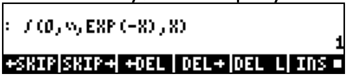
↱ EQW ↱ ∫ 0 ▶ ↰ ∞ ▶ ↰ $e^x$ +/- X ▶ X ENTER

In Algebraic mode, the following screen shows the result of these keystrokes with neither *_Small* nor *_Textbook* are selected:



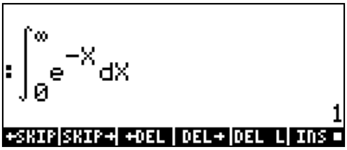
```
: ∫(0,∞,EXP(-X),X)
                                1
←SKIP SKIP→ ←DEL  DEL→ DEL L INS ■
```

With the *_Small* option selected only, the display looks as shown below:



```
: ∫(0,∞,EXP(-X),X)
                                1
←SKIP SKIP→ ←DEL  DEL→ DEL L INS ■
```

With the *_Textbook* option selected (default value), regardless of whether the *_Small* option is selected or not, the display shows the following result:



: $\int_0^\infty e^{-X} dX$

1

←SKIP SKIP→ ←DEL DEL→ DEL L INS ■

## Selecting properties of the equation writer (EQW)

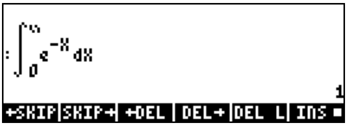
First, press the MODE button to activate the CALCULATOR MODES input form. Within the CALCULATOR MODES input form, press the DISP soft menu key (F4) to display the DISPLAY MODES input form. Press the down arrow key, ▼, three times, to get to the *EQW* (Equation Writer) line. This line shows two properties that can be modified. When these properties are selected (checked) the following effects are activated:

*_Small* Changes font size to small while using the equation editor

*_Small Stack Disp* Shows small font in the stack for textbook style display

Detailed instructions on the use of the equation editor (EQW) are presented elsewhere in this manual.

For the example of the integral $\int_0^{\infty} e^{-X}\,dX$, presented above, selecting the *_Small Stack Disp* in the *EQW* line of the DISPLAY MODES input form produces the following display:



## Selecting the size of the header

First, press the MODE button to activate the CALCULATOR MODES input form. Within the CALCULATOR MODES input form, press the DISP soft menu key (F4) to display the DISPLAY MODES input form. Press the down arrow key, ▼, four times, to get to the *Header* line. The value *2* is assigned to the *Header* field by default. This means that the top part of the display will contain two lines, one showing the current settings of the calculator, and a second one showing the current sub directory within the calculator's memory (These lines were described earlier in the manual). The user can select to change this setting to *1* or *0* to reduce the number of header lines in the display.

## Selecting the clock display

First, press the MODE button to activate the CALCULATOR MODES input form. Within the CALCULATOR MODES input form, press the DISP soft menu key (F4) to display the DISPLAY MODES input form. Press the down arrow key, ▼, four times, to get to the *Header* line. The *Header* field will be highlighted. Use the right arrow key (▶) to select the underline in front of the options *_Clock* or *_Analog*. Toggle the CHK soft menu key until the desired setting is achieved. If the *_Clock* option is selected, the time of the day and date will be shown in the upper right corner of the display. If the *_Analog* option is also selected, an analog clock, rather than a digital clock, will be shown in the upper right corner of the display. If the *_Clock* option is not selected, or the header is not present, or too small, the date and time of day will not be shown in the display.

# Chapter 2
# Introducing the calculator

In this chapter we present a number of basic operations of the calculator including the use of the Equation Writer and the manipulation of data objects in the calculator. Study the examples in this chapter to get a good grasp of the capabilities of the calculator for future applications.

## Calculator objects

Any number, expression, character, variable, etc., that can be created and manipulated in the calculator is referred to as an object. Some of the most useful type of objects are listed below.

**Real**. These object represents a number, positive or negative, with 12 significant digits and an exponent ranging from -499 to +499. example of reals are: 1., -5., 56.41564 1.5E45, -555.74E-95

When entering a real number, you can use the EEX key to enter the exponent and the +/- key to change the sign of the exponent or mantissa.

Note that real must be entered with a decimal point, even if the number does not have a fractional part. Otherwise the number is taken as an integer number, which is a different calculator objects. Reals behave as you would expect a number to when used in a mathematical operation.

**Integers**. These objects represent integer numbers (numbers without fractional part) and do not have limits (except the memory of the calculator). Example of integers are: 1, 564654112, -413165467354646765465487. Note how these numbers do not have a decimal point.

Due to their storage format, integer numbers are always maintain full precision in their calculation. For example, an operation such as 30/14, with integer numbers, will return 15/7 and not 2.142.... To force a real (or floating-point) result, use function →NUM ↱ →NUM .

Integers are used frequently in CAS-based functions as they are designed to keep full precision in their operation..

If the approximate mode (APPROX) is selected in the CAS (see Appendix C), integers will be automatically converted to reals. If you are not planning to use the CAS, it might be a good idea to switch directly into approximate mode. Refer to Appendix C for more details.

Mixing integers and reals together or mistaking an integer for a real is a common occurrence. The calculator will detect such mixing of objects and ask you if you want to switch to approximate mode.

**Complex numbers**, are an extension of real numbers that include the unit imaginary number, $i^2$= -1. A complex number, e.g., *3 + 2i*, is written as *(3, 2)* in the calculator.

Complex numbers can be displayed in either Cartesian or polar mode depending on the setting selected. Note that complex numbers are always stored in Cartesian mode and that only the display is affected. This allows the calculator to keep as much precision as possible during calculations.

Most mathematics functions work on complex numbers. There is no need to use a special "complex +" function to add complex numbers, you can use the same + function that on reals or integers.

Vector and matrix operations utilize objects of type 3, **real arrays**, and, if needed, type 4, **complex arrays**. Objects type 2, **strings**, are simply lines of text (enclosed between quotes) produced with the alphanumeric keyboard.

A **list** is just a collection of objects enclosed between curly brackets and separated by spaces in RPN mode (the space key is labeled SPC), or by commas in algebraic mode. Lists, objects of type 5, can be very useful when processing collections of numbers. For example, the columns of a table can be entered as lists. If preferred, a table can be entered as a matrix or array.

Objects type 8 are *programs in User RPL language*. These are simply sets of instructions enclosed between the symbols << >>.

Associated with programs are objects types 6 and 7, **Global** and **Local Names**, respectively. These names, or variables, are used to store any type of objects. The concept of global or local names is related to the scope or reach of the variable in a given program.

An **algebraic object**, or simply, an **algebraic** (object of type 9), is a valid algebraic expression enclosed between apostrophes.

**Binary integers**, objects of type 10, are used in some computer science applications.

**Graphics objects**, objects of type 11, store graphics produced by the calculator.

**Tagged objects**, objects of type 12, are used in the output of many programs to identify results. For example, in the tagged object: Mean: 23.2, the word Mean: is the tag used to identify the number 23.2 as the mean of a sample, for example.

**Unit objects**, objects of type 13, are numerical values with a physical unit attached to them.

**Directories**, objects of type 15, are memory locations used to organize your variables in a similar fashion as folders are used in a personal computer.

**Libraries**, objects of type 16, are programs residing in memory ports that are accessible within any directory (or sub-directory) in your calculator. They resemble *built-in functions*, objects of type 18, and *built-in commands*, objects of type 19, in the way they are used.

# Editing expressions in the screen

In this section we present examples of expression editing directly into the calculator display (algebraic history or RPN stack).

## Creating arithmetic expressions

For this example, we select the Algebraic operating mode and select a *Fix* format with 3 decimals for the display. We are going to enter the arithmetic expression:

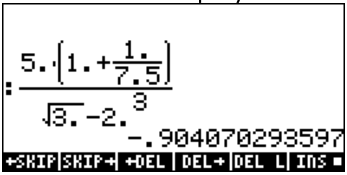
$$5.0 \cdot \frac{1.0 + \frac{1.0}{7.5}}{\sqrt{3.0} - 2.0^3}$$

To enter this expression use the following keystrokes:

5 • × ↰() 1 • + 1 • ÷ 7 • 5 ▶ ÷
↰() √x 3 • − 2 • yˣ 3

The resulting expression is: 5.*(1.+1./7.5)/(√3.-2.^3).

Press ENTER to get the expression in the display as follows:



$$:\frac{5.\cdot\left(1.+\frac{1.}{7.5}\right)}{\sqrt{3.}-2.^{3}}$$

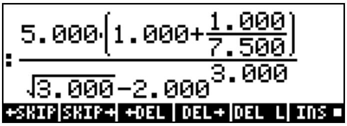
-.904070293597

←SKIP SKIP→ ←DEL DEL→ DEL L INS ■

Notice that, if your CAS is set to EXACT (see Appendix C) and you enter your expression using integer numbers for integer values, the result is a symbolic quantity, e.g.,

5 × ↰() 1 + 1 ÷ 7 • 5 ▶ ÷
↰() √x 3 − 2 yˣ 3

Before producing a result, you will be asked to change to Approximate mode. Accept the change to get the following result (shown in Fix decimal mode with three decimal places – see Chapter 1):



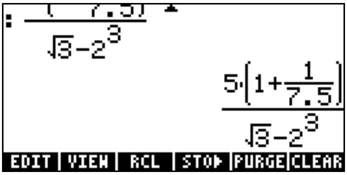
$$:\frac{5.000\cdot\left(1.000+\frac{1.000}{7.500}\right)}{\sqrt{3.000}-2.000^{3.000}}$$

←SKIP SKIP→ ←DEL DEL→ DEL L INS ■

In this case, when the expression is entered directly into the stack, as soon as you press ENTER, the calculator will attempt to calculate a value for the expression. If the expression is entered between quotes, however, the calculator will reproduce the expression as entered. In the following example, we enter the same expression as above, but using quotes. For this case we set the operating mode to Algebraic, the CAS mode to Exact (deselect _*Approx*), and the display setting to *Textbook*. The keystrokes to enter the expression are the following:

' 5 × ↰() 1 + 1 ÷ 7 • 5 ▶ ÷
↰() √x 3 − 2 yˣ 3 ENTER

The result will be shown as follows:



$$\frac{5\cdot\left(1+\frac{1}{7.5}\right)}{\sqrt{3}-2^{3}}$$

To evaluate the expression we can use the EVAL function, as follows:

↱ EVAL ↰ ANS

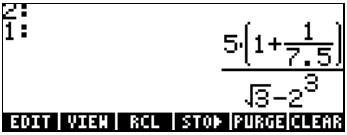
As in the previous example, you will be asked to approve changing the CAS setting to *Approx*. Once this is done, you will get the same result as before.

An alternative way to evaluate the expression entered earlier between quotes is by using the option ↱ →NUM . To recover the expression from the existing stack, use the following keystrokes: ◀ ◀ ↱ →NUM

We will now enter the expression used above when the calculator is set to the RPN operating mode. We also set the CAS to *Exact* and the display to *Textbook*. The keystrokes to enter the expression between quotes are the same used earlier, i.e.,

' 5 × ↰ () 1 + 1 ÷ 7 . 5 ▶ ÷
↰ () √x 3 − 2 yˣ 3 ENTER

Resulting in the output



2:
1:
$$\frac{5\cdot\left(1+\frac{1}{7.5}\right)}{\sqrt{3}-2^{3}}$$

Press ENTER once more to keep two copies of the expression available in the stack for evaluation. We first evaluate the expression using the function *EVAL*, and next using the function *→NUM*. First, evaluate the expression using function EVAL. The resulting expression is semi-symbolic in the sense that there are floating-point components to the result, as well as a √3. Next, we switch stack locations and evaluate using function →NUM:

▶ Exchange stack levels 1 and 2 (the SWAP command)

↱ →NUM Evaluate using function →NUM

This latter result is purely numerical, so that the two results in the stack, although representing the same expression, seem different. To verify that they are not, we subtract the two values and evaluate this difference using function EVAL:

−  Subtract level 1 from level 2

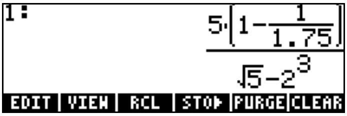
↱ EVAL  Evaluate using function EVAL

The result is zero (0.).

**Note**: Avoid mixing integer and real data to avoid conflicts in the calculations. For many physical science and engineering applications, including numerical solution of equation, statistics applications, etc., the APPROX mode (see Appendix C) works better. For mathematical applications, e.g., calculus, vector analysis, algebra, etc., the EXACT mode is preferred. Become acquainted with operations in both modes and learn how to switch from one to the other for different types of operations (see Appendix C).

## Editing arithmetic expressions

Suppose that we entered the following expression, between quotes, with the calculator in RPN mode and the CAS set to EXACT:

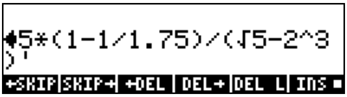


1: $\dfrac{5\cdot\left(1-\dfrac{1}{1.75}\right)}{\sqrt{5}-2^{3}}$

rather than the intended expression: $5\cdot\dfrac{1+\dfrac{1}{7.5}}{\sqrt{3}-2^{3}}$. The incorrect expression was entered by using:

' 5 × ↰ ()  1 + 1 ÷ 1 . 7 5 ▶ ÷ ↰ ()
√x 5 − 2 yˣ 3 ENTER

To enter the line editor use ↰ ▼. The display now looks as follows:



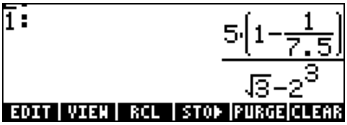
```
'5*(1-1/1.75)/(√5-2^3
)'
```

The editing cursor is shown as a blinking left arrow over the first character in the line to be edited. Since the editing in this case consists of removing some characters and replacing them with others, we will use the right and left arrow keys, ◀ ▶, to move the cursor to the appropriate place for editing, and the delete key, ⬅, to eliminate characters.

The following keystrokes will complete the editing for this case:

- Press the right arrow key, ▶, until the cursor is immediately to the right of the decimal point in the term *1.75*
- Press the delete key, ⬅, twice to erase the characters *1.*
- Press the right arrow key, ▶, once, to move the cursor to the right of the *7*
- Type a decimal point with .
- Press the right arrow key, ▶, until the cursor is immediately to the right of the *5*
- Press the delete key, ⬅, once to erase the character *5*
- Type a *3* with *3*
- Press ENTER to return to the stack

The edited expression is now available in the stack.



1: $\frac{5\cdot\left(1-\frac{1}{7.5}\right)}{\sqrt{3}-2^{3}}$

Editing of a line of input when the calculator is in Algebraic operating mode is exactly the same as in the RPN mode. You can repeat this example in Algebraic mode to verify this assertion.

## Creating algebraic expressions

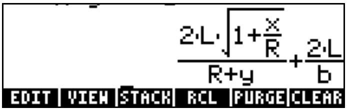
Algebraic expressions include not only numbers, but also variable names. As an example, we will enter the following algebraic expression:

$$\frac{2L\sqrt{1+\frac{x}{R}}}{R+y}+2\frac{L}{b}$$

We set the calculator operating mode to Algebraic, the CAS to *Exact*, and the display to *Textbook*. To enter this algebraic expression we use the following keystrokes:

↱ ' 2 × ALPHA L × √x ↰ () / + ALPHA ↰ X ÷ ALPHA R ▶
÷ ↰ () ALPHA R + ALPHA ↰ Y ▶ + 2 × ALPHA L ÷ ALPHA ↰ B

Press ENTER to get the following result:



$$\frac{2\cdot L\cdot\sqrt{1+\frac{x}{R}}}{R+y}+\frac{2\cdot L}{b}$$

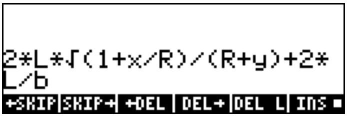
Entering this expression when the calculator is set in the RPN mode is exactly the same as this Algebraic mode exercise.

## Editing algebraic expressions

Editing of an algebraic expression with the line editor is very similar to that of an arithmetic expression (see exercise above). Suppose that we want to modify the expression entered above to read

$$\frac{2L\sqrt{1+\frac{x^2}{R}}}{R+x}+2\sqrt{\frac{L}{b}}$$

To edit this algebraic expression using the line editor use ↰ ▼ . This activates the line editor, showing the expression to be edited as follows:



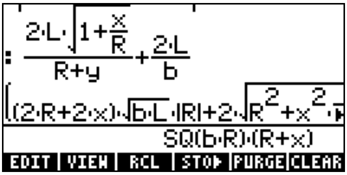
```
2*L*√(1+x/R)/(R+y)+2*
L/b
```

The editing cursor is shown as a blinking left arrow over the first character in the line to be edited. As in an earlier exercise on line editing, we will use the right and left arrow keys, ◀▶, to move the cursor to the appropriate place for editing, and the delete key, ⬅, to eliminate characters.

The following keystrokes will complete the editing for this case:

- Press the right arrow key, ▶, until the cursor is to the right of the *x*
- Type $Y^x$ 2 to enter the power 2 for the *x*
- Press the right arrow key, ▶, until the cursor is to the right of the *y*
- Press the delete key, ⬅, once to erase the characters *y.*
- Type ALPHA ↰ X to enter an *x*
- Press the right arrow key, ▶, 4 times to move the cursor to the right of the *
- Type $\sqrt{x}$ to enter a square root symbol
- Type ↰ () to enter a set of parentheses (they come in pairs)
- Press the right arrow key, ▶, once, and the delete key, ⬅, once, to delete the right parenthesis of the set inserted above
- Press the right arrow key, ▶, 4 times to move the cursor to the right of the *b*
- Type ↰ () to enter a second set of parentheses
- Press the delete key, ⬅, once, to delete the left parenthesis of the set inserted above.
- Press ENTER to return to normal calculator display.

The result is shown next:

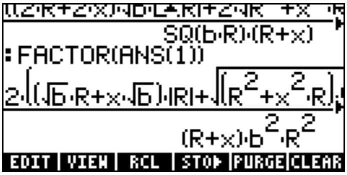


$$:\frac{2\cdot L\cdot\sqrt{1+\frac{x}{R}}}{R+y}+\frac{2\cdot L}{b}$$

$$((2\cdot R+2\cdot x)\cdot\sqrt{b\cdot L}\cdot|R|+2\cdot\sqrt{R^2+x^2\cdot}$$

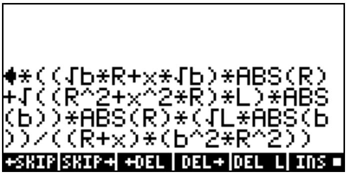
SQ(b·R)·(R+x)

Notice that the expression has been expanded to include terms such as |R|, the absolute value, and SQ(b·R), the square of b·R. To see if we can simplify this result, use FACTOR(ANS(1)) in ALG mode:



```
SQ(b·R)·(R+x)
:FACTOR(ANS(1))
2·((√b·R+x·√b)·|R|+√((R^2+x^2·R)
(R+x)·b^2·R^2
```

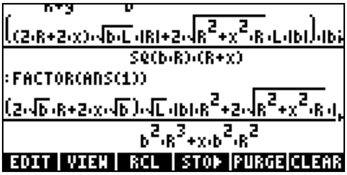
- Press ⬑ ▼ to activate the line editor once more. The result is now:



```
◆*((√b*R+x*√b)*ABS(R)
+√((R^2+x^2*R)*L)*ABS
(b))*ABS(R)*(√L*ABS(b
))/((R+x)*(b^2*R^2))
```

- Pressing ENTER once more to return to normal display.

To see the entire expression in the screen, we can change the option *_Small Stack Disp* in the *DISPLAY MODES* input form (see Chapter 1). After effecting this change, the display will look as follows:



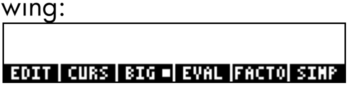
```
((2·R+2·x)·√(b·L)·|R|+2·√(R^2+x^2·R)·L·|b|)·|b|
SQ(b·R)·(R+x)
:FACTOR(ANS(1))
(2·√b·R+2·x·√b)·√L·|b|·R^2+2·√(R^2+x^2·R)·|
b^2·R^3+x·b^2·R^2
```

**Note**: To use Greek letters and other characters in algebraic expressions use the CHARS menu. This menu is activated by the keystroke combination ⮣ CHARS . Details are presented in Appendix D.

## Using the Equation Writer (EQW) to create expressions

The equation writer is an extremely powerful tool that not only let you enter or see an equation, but also allows you to modify and work/apply functions on all or part of the equation. The equation writer (EQW), therefore, allows you to perform complex mathematical operations, directly, or in a step-by-step mode, as you would do on paper when solving, for example, calculus problems.

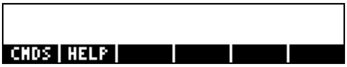
The Equation Writer is launched by pressing the keystroke combination ↱ ↱ EQW (the third key in the fourth row from the top in the keyboard). The resulting screen is the following:



The six soft menu keys for the Equation Writer activate the following functions:

EDIT: lets the user edit an entry in the line editor (see examples above)
CURS: highlights expression and adds a graphics cursor to it
BIG: if selected (selection shown by the character in the label) the font used in the writer is the system font 8 (the largest font available)
EVAL: lets you evaluate, symbolically or numerically, an expression highlighted in the equation writer screen (similar to ↱ EVAL)
FACTO: lets you factor an expression highlighted in the equation writer screen (if factoring is possible)
SIMP: lets you simplify an expression highlighted in the equation writer screen (as much as it can be simplified according to the algebraic rules of the CAS)

If you press the NXT key, two more soft menu options show up as shown below:



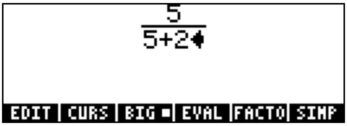
The six soft menu keys for the Equation Writer activate the following functions:

CMDS: allows access to the collection of CAS commands listed in alphabetical order. This is useful to insert CAS commands in an expression available in the Equation Writer.
HELP: activates the calculator's CAS help facility to provide information and examples of CAS commands.
Some examples for the use of the Equation Writer are shown below.

## Creating arithmetic expressions

Entering arithmetic expressions in the Equation Writer is very similar to entering an arithmetic expression in the stack enclosed in quotes. The main difference is that in the Equation Writer the expressions produced are written in "textbook" style instead of a line-entry style. Thus, when a division sign (i.e., ÷) is entered in the Equation Writer, a fraction is generated and the cursor placed in the numerator. To move to the denominator you must use the down arrow key. For example, try the following keystrokes in the Equation Writer screen: 5 ÷ 5 + 2

The result is the expression

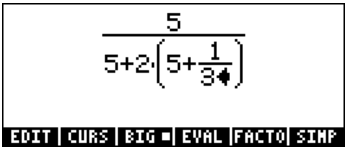


$$\frac{5}{5+2}$$

The cursor is shown as a left-facing key. The cursor indicates the current edition location. Typing a character, function name, or operation will enter the corresponding character or characters in the cursor location. For example, for the cursor in the location indicated above, type now:

× ↰ () 5 + 1 ÷ 3

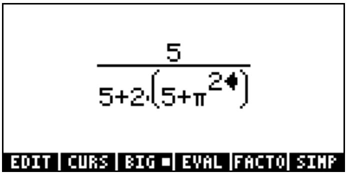
The edited expression looks as follows:



$$\frac{5}{5+2\cdot\left(5+\frac{1}{3}\right)}$$

Suppose that you want to replace the quantity between parentheses in the denominator (i.e., 5+1/3) with (5+$\pi^2$/2). First, we use the delete key (◀) delete the current 1/3 expression, and then we replace that fraction with $\pi^2$/2, as follows: ◀ ◀ ◀ ↰ π $Y^x$ 2

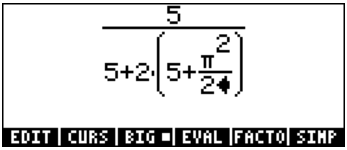
When we hit this point the screen looks as follows:



$$\frac{5}{5+2\cdot\left(5+\pi^{2}\right)}$$

In order to insert the denominator *2* in the expression, we need to highlight the entire $\pi^2$ expression. We do this by pressing the right arrow key (▶) once. At that point, we enter the following keystrokes: ÷ 2

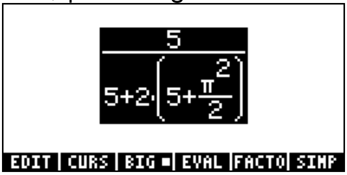
The expression now looks as follows:



Suppose that now you want to add the fraction 1/3 to this entire expression, i.e., you want to enter the expression:

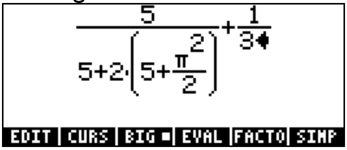
$$\frac{5}{5+2\cdot(5+\frac{\pi^2}{2})}+\frac{1}{3}$$

First, we need to highlight the entire first term by using either the right arrow (▶) or the upper arrow (▲) keys, repeatedly, until the entire expression is highlighted, i.e., seven times, producing:



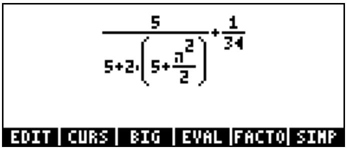
**NOTE**: Alternatively, from the original position of the cursor (to the right of the 2 in the denominator of $\pi^2/2$), we can use the keystroke combination ↱▲ , interpreted as (↱ ⤒ ).

Once the expression is highlighted as shown above, type + 1 ÷ 3 to add the fraction 1/3. Resulting in:



### Showing the expression in smaller-size

To show the expression in a smaller-size font (which could be useful if the expression is long and convoluted), simply press the BIG F3 soft menu key. For this case, the screen looks as follows:



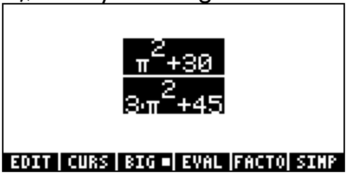
$$\frac{5}{5+2\cdot\left(5+\frac{\pi^2}{2}\right)}+\frac{1}{3}$$

To recover the larger-font display, press the BIG F3 soft menu key once more.

### Evaluating the expression

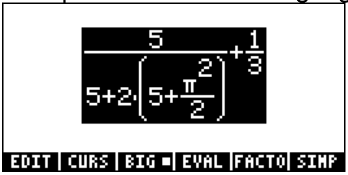
To evaluate the expression (or parts of the expression) within the Equation Writer, highlight the part that you want to evaluate and press the EVAL F4 soft menu key.

For example, to evaluate the entire expression in this exercise, first, highlight the entire expression, by pressing ↱ ▲. Then, press the EVAL F4 soft menu key. If your calculator is set to Exact CAS mode (i.e., the *_Approx* CAS mode is not checked), then you will get the following symbolic result:



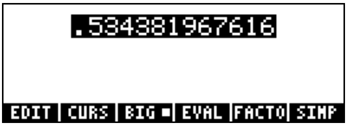
$$\frac{\pi^2+30}{3\cdot\pi^2+45}$$

If you want to recover the unevaluated expression at this time, use the function UNDO, i.e., ↱ UNDO (the first key in the third row of keys from the top of the keyboard). The recovered expression is shown highlighted as before:



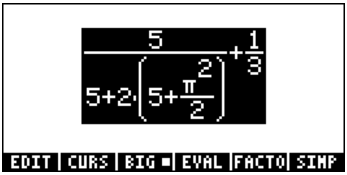
$$\frac{5}{5+2\cdot\left(5+\frac{\pi^2}{2}\right)}+\frac{1}{3}$$

If you want a floating-point (numerical) evaluation, use the *→NUM* function (i.e., ↱ →NUM ). The result is as follows:



.534381967616

Use the function UNDO ( ↱ UNDO ) once more to recover the original expression:

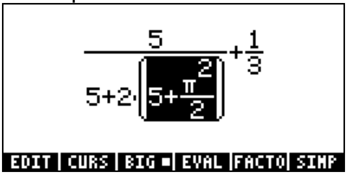


$$\frac{5}{5+2\cdot\left(5+\frac{\pi^2}{2}\right)}+\frac{1}{3}$$

### Evaluating a sub-expression

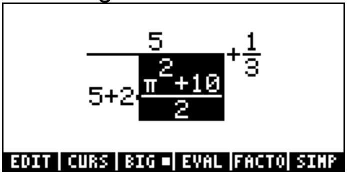
Suppose that you want to evaluate only the expression in parentheses in the denominator of the first fraction in the expression above. You have to use the arrow keys to select that particular sub-expression. Here is a way to do it:

▼ Highlights only the first fraction
▼ Highlights the numerator of the first fraction
▶ Highlights denominator of the first fraction
▼ Highlights first term in denominator of first fraction
▶ Highlights second term in denominator of first fraction
▼ Highlights first factor in second term in denominator of first fraction
▶ Highlights expression in parentheses in denominator of first fraction



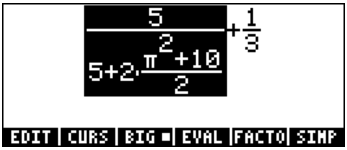
$$\frac{5}{5+2\cdot\left(5+\frac{\pi^2}{2}\right)}+\frac{1}{3}$$

Since this is the sub-expression we want evaluated, we can now press the EVAL F4 soft menu key, resulting in:



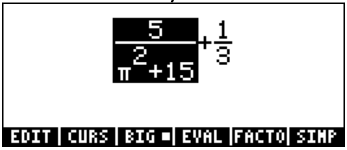
$$\frac{5}{5+2\cdot\frac{\pi^2+10}{2}}+\frac{1}{3}$$

A symbolic evaluation once more. Suppose that, at this point, we want to evaluate the left-hand side fraction only. Press the upper arrow key (▲) three times to select that fraction, resulting in:



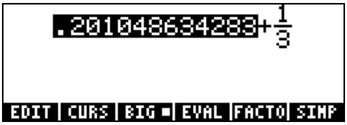
$\frac{5}{5+2\cdot\frac{\pi^2+10}{2}}+\frac{1}{3}$

Then, press the EVAL F4 soft menu key to obtain:



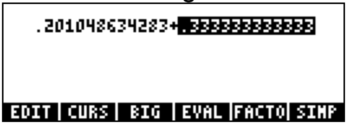
$\frac{5}{\pi^2+15}+\frac{1}{3}$

Let's try a numerical evaluation of this term at this point. Use ↱ →NUM to obtain:



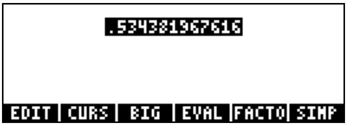
$.201048634283+\frac{1}{3}$

Let's highlight the fraction to the right, and obtain a numerical evaluation of that term too, and show the sum of these two decimal values in small-font format by using:▶ ↱ →NUM F3 , we get:



.201048634283+.333333333333

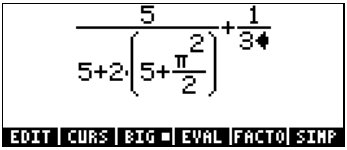
To highlight and evaluate the expression in the Equation Writer we use: ▲ F4 , resulting in:



.534381967616

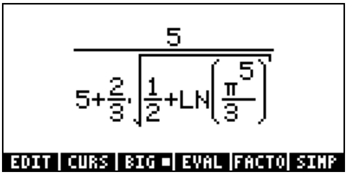
## Editing arithmetic expressions

We will show some of the editing features in the Equation Writer as an exercise. We start by entering the following expression used in the previous exercises:



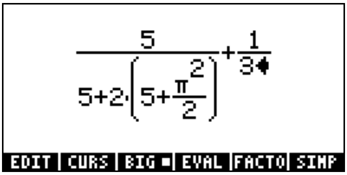
$\frac{5}{5+2\cdot\left(5+\frac{\pi^2}{2}\right)}+\frac{1}{3}$

And will use the editing features of the Equation Editor to transform it into the following expression:



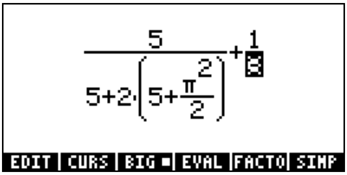
$$\frac{5}{5+\frac{2}{3}\cdot\sqrt{\frac{1}{2}+LN\left(\frac{\pi^{5}}{3}\right)}}$$

In the previous exercises we used the arrow keys to highlight sub-expressions for evaluation. In this case, we will use them to trigger a special editing cursor. After you have finished entering the original expression, the typing cursor (a left-pointing arrow) will be located to the right of the 3 in the denominator of the second fraction as shown here:



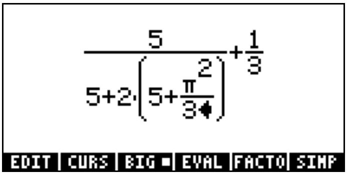
$$\frac{5}{5+2\cdot\left(5+\frac{\pi^{2}}{2}\right)}+\frac{1}{3}$$

Press the down arrow key (▼) to trigger the clear editing cursor. The screen now looks like this:



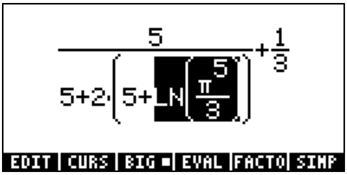
$$\frac{5}{5+2\cdot\left(5+\frac{\pi^{2}}{2}\right)}+\frac{1}{3}$$

By using the left arrow key (◀) you can move the cursor in the general left direction, but stopping at each individual component of the expression. For example, suppose that we will first will transform the expression *$\pi^2/2$* into the expression *LN($\pi^5/3$)* . With the clear cursor active, as shown above, press the left-arrow key (◀) twice to highlight the *2* in the denominator of *$\pi^2/2$*. Next, press the delete key (⬅) once to change the cursor into the insertion cursor. Press ⬅ once more to delete the 2, and then 3 to enter a 3. At this point, the screen looks as follows:



$$\frac{5}{5+2\cdot\left(5+\frac{\pi^{2}}{3}\right)}+\frac{1}{3}$$

Next, press the down arrow key (▼) to trigger the clear editing cursor highlighting the 3 in the denominator of $\pi^2/3$. Press the left arrow key (◄) once to highlight the exponent 2 in the expression $\pi^2/3$. Next, press the delete key (⇐) once to change the cursor into the insertion cursor. Press ⇐ once more to delete the 2, and then 5 to enter a 5. Press the upper arrow key (▲) three times to highlight the expression $\pi^5/3$. Then, type ↱ LN to apply the *LN* function to this expression. The screen now looks like this:



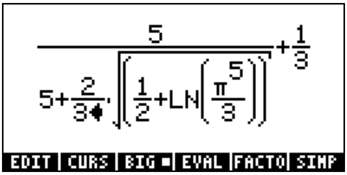
$$\frac{5}{5+2\cdot\left(5+LN\left(\frac{\pi^5}{3}\right)\right)}+\frac{1}{3}$$

Next, we'll change the 5 within the parentheses to a ½ by using these keystrokes: ◄ ⇐ ⇐ 1 ÷ 2

Next, we highlight the entire expression in parentheses an insert the square root symbol by using: ▲ ▲ ▲ ▲ √x

Next, we'll convert the 2 in front of the parentheses in the denominator into a 2/3 by using: ◄ ⇐ ⇐ 2 ÷ 3

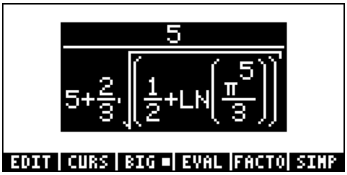
At this point the expression looks as follows:



$$\frac{5}{5+\frac{2}{3}\cdot\sqrt{\left(\frac{1}{2}+LN\left(\frac{\pi^5}{3}\right)\right)}}+\frac{1}{3}$$

The final step is to remove the 1/3 in the right-hand side of the expression. This is accomplished by using: ▲ ▲ ▲ ▲ ▲ ► ⇐ ⇐ ⇐ ⇐ ⇐

The final version will be:



$$\frac{5}{5+\frac{2}{3}\cdot\sqrt{\left(\frac{1}{2}+LN\left(\frac{\pi^5}{3}\right)\right)}}$$

In summary, to edit an expression in the Equation Writer you should use the arrow keys (◄ ► ▲ ▼) to highlight expression to which functions will be applied (e.g., the *LN* and square root cases in the expression above). Use the down arrow key (▼) in any location, repeatedly, to trigger the clear editing cursor. In this mode, use the left or right arrow keys (◀ ▶) to move from term to term in an expression. When you reach a point that you need to edit, use the delete key (◀) to trigger the insertion cursor and proceed with the edition of the expression.

## Creating algebraic expressions

An algebraic expression is very similar to an arithmetic expression, except that English and Greek letters may be included. The process of creating an algebraic expression, therefore, follows the same idea as that of creating an arithmetic expression, except that use of the alphabetic keyboard is included.

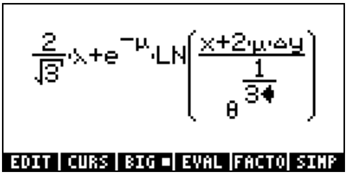
To illustrate the use of the Equation Writer to enter an algebraic equation we will use the following example. Suppose that we want to enter the expression:

$$\frac{2}{\sqrt{3}}\lambda + e^{-\mu} \cdot LN\left(\frac{x+2\mu\cdot\Delta y}{\theta^{1/3}}\right)$$

Use the following keystrokes:

2 ÷ √X 3 ▶ ▶ × ALPHA ↱ M + ↰ e^x +/- ALPHA ↱ M
▶ ▶ × ↱ LN ALPHA ↰ X + 2 × ALPHA ↱ M × ALPHA ↱ C
ALPHA ↰ Y ▲ ▲ ▲ ÷ ALPHA ↱ T Y^x 1 ÷ 3

This results in the output:


$$\frac{2}{\sqrt{3}}\cdot\lambda+e^{-\mu}\cdot LN\left(\frac{x+2\cdot\mu\cdot\Delta y}{\theta^{\frac{1}{3}}}\right)$$

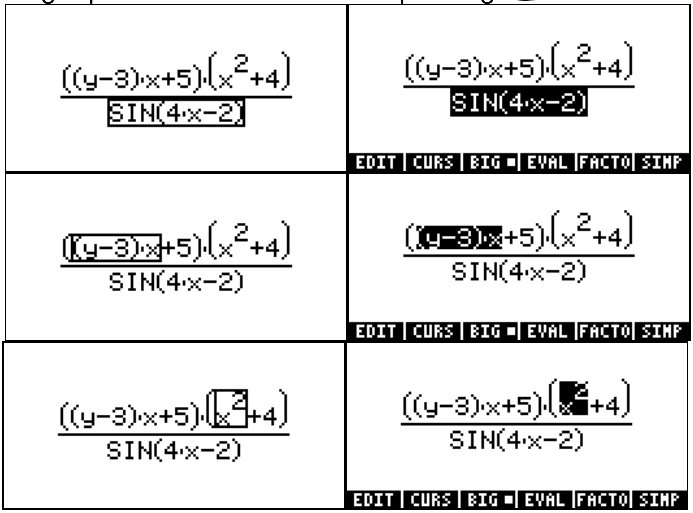
In this example we used several lower-case English letters, e.g., x (ALPHA ↰ X), several Greek letters, e.g., *λ* (ALPHA ↱ N), and even a combination of Greek and English letters, namely, *Δy* (ALPHA ↱ C ALPHA ↰ Y). Keep in mind that to enter a lower-case English letter, you need to use the combination: ALPHA ↰ followed by the letter you want to enter. Also, you can always copy special characters by using the CHARS menu (↱ CHARS) if you don't want to memorize the keystroke combination that produces it. A listing of commonly used ALPHA ↱ keystroke combinations was listed in an earlier section.

### The expression tree

The expression tree is a diagram showing how the Equation Writer interprets an expression. See Appendix E for a detailed example.

### The CURS function

The CURS function (CURS) in the Equation Writer menu (the F2 key) converts the display into a graphical display and produces a graphical cursor that can be controlled with the arrow keys (◀▶▲▼) for selecting sub-expressions. The sub-expression selected with CURS will be shown framed in the graphics display. After selecting a sub-expression you can press ENTER to show the selected sub-expression highlighted in the Equation writer. The following figures show different sub-expressions selected with and the corresponding Equation Writer screen after pressing ENTER.



$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\boxed{\mathrm{SIN}(4\cdot x-2)}}$$

$$\frac{(\boxed{(y-3)\cdot x}+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

$$\frac{((y-3)\cdot x+5)\cdot(\boxed{x^2}+4)}{\mathrm{SIN}(4\cdot x-2)}$$

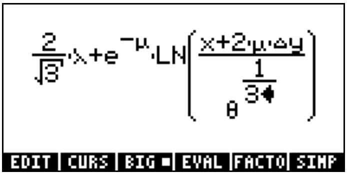
## Editing algebraic expressions

The editing of algebraic equations follows the same rules as the editing of algebraic equations. Namely:

- Use the arrow keys (◀▶▲▼) to highlight expressions
- Use the down arrow key (▼), repeatedly, to trigger the clear editing cursor . In this mode, use the left or right arrow keys (◀▶) to move from term to term in an expression.

- At an editing point, use the delete key (⌫) to trigger the insertion cursor and proceed with the edition of the expression.

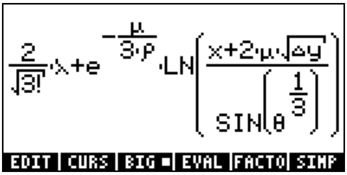
To see the clear editing cursor in action, let's start with the algebraic expression that we entered in the exercise above:



Press the down arrow key, ▼, at its current location to trigger the clear editing cursor. The 3 in the exponent of *θ* will be highlighted. Use the left arrow key, ◄, to move from element to element in the expression. The order of selection of the clear editing cursor in this example is the following (press the left arrow key, ◄, repeatedly):

1. The 1 in the 1/3 exponent
2. θ
3. Δy
4. μ
5. 2
6. x
7. μ in the exponential function
8. λ
9. 3 in the √3 term
10. the 2 in the 2/√3 fraction

At any point we can change the clear editing cursor into the insertion cursor by pressing the delete key (⌫). Let's use these two cursors (the clear editing cursor and the insertion cursor) to change the current expression into the following:



If you followed the exercise immediately above, you should have the clear editing cursor on the number *2* in the first factor in the expression. Follow these keystrokes to edit the expression:

▶ ALPHA ↱ 2 Enters the factorial for the 3 in the square root (entering the factorial changes the cursor to the selection cursor)

▼ ▼ ▶ ▶ Selects the $\mu$ in the exponential function

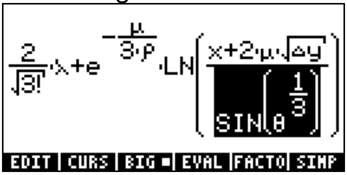
÷ 3 × ALPHA ↱ F Modifies exponential function argument

▶ ▶ ▶ ▶ Selects *Δy*

√X Places a square root symbol on *Δy* (this operation also changes the cursor to the selection cursor)

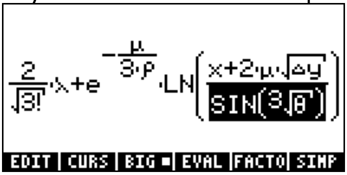
▼ ▼ ▶ ▲ ▲ SIN Select $\theta^{1/3}$ and enter the SIN function

The resulting screen is the following:

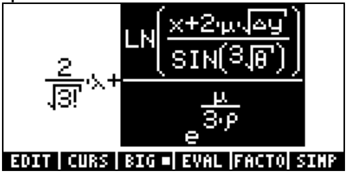


## Evaluating a sub-expression

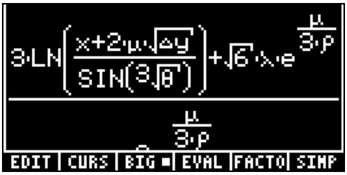
Since we already have the sub-expression $SIN(\theta^{1/3})$ highlighted, let's press the EVAL F4 soft menu key to evaluate this sub-expression. The result is:



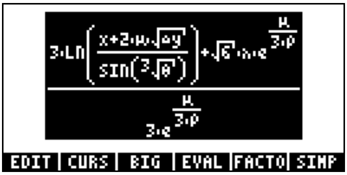
Some algebraic expressions cannot be simplified anymore. Try the following keystrokes: ▲ F4 . You will notice that nothing happens, other than the highlighting of the entire argument of the *LN* function. This is because this expression cannot be evaluated (or simplified) any more according to the CAS rules. Trying the keystrokes: ▲ F4 again does not produce any changes on the expression. Another sequence of ▲ F4 keystrokes, however, modifies the expression as follows:



One more application of the ▲ F4 keystrokes produces more changes:



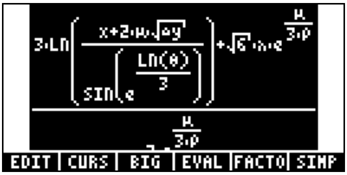
This expression does not fit in the Equation Writer screen. We can see the entire expression by using a smaller-size font. Press the BIG F3 soft menu key to get:



Even with the larger-size font, it is possible to navigate through the entire expression by using the clear editing cursor. Try the following keystroke sequence: F3 ▼▼▼▼, to set the clear editing cursor atop the factor 3 in the first term of the numerator. Then, press the right arrow key, ▶, to navigate through the expression.

**Simplifying an expression**

Press the BIG F3 soft menu key to get the screen to look as in the previous figure (see above). Now, press the SIMP F3 soft menu key, to see if it is possible to simplify this expression as it is shown in the Equation Writer. The result is the following screen:



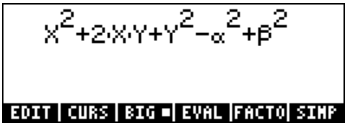
This screen shows the argument of the SIN function, namely, $\sqrt[3]{\theta}$, transformed into $e^{\frac{LN(\theta)}{3}}$. This may not seem like a simplification, but it is in the sense that the cubic root function has been replaced by the inverse functions exp-LN.

### Factoring an expression

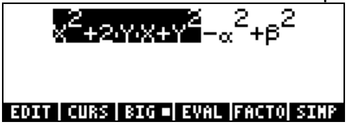
In this exercise we will try factoring a polynomial expression. To continue the previous exercise, press the ENTER key. Then, launch the Equation Writer again by pressing the ↱ EQW key. Type the equation:

X y^x 2 ▶ + 2 × X × ALPHA Y + ALPHA Y y^x 2 ▶ −
ALPHA ↱ A y^x 2 ▶ ▶ + ALPHA ↱ B y^x 2

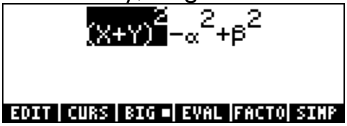
resulting in


$X^2+2\cdot X\cdot Y+Y^2-\alpha^2+\beta^2$

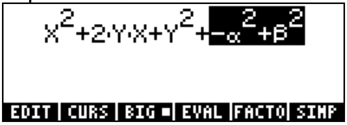
Let's select the first 3 terms in the expression and attempt a factoring of this sub-expression: ↱ ▲ ▼ ↱ ▶ ↱ ▶ . This produces:


$X^2+2\cdot Y\cdot X+Y^2-\alpha^2+\beta^2$

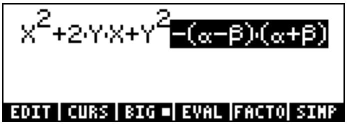
Now, press the FACTO soft menu key, to get


$(X+Y)^2-\alpha^2+\beta^2$

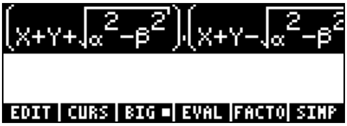
Press ↱ UNDO to recover the original expression. Next, enter the following keystrokes: ▼ ▼ ▼ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▲ ▲ ▲ ↱ ▶ to select the last two terms in the expression, i.e.,


$X^2+2\cdot Y\cdot X+Y^2+-\alpha^2+\beta^2$

press the FACTO soft menu key, to get


$X^2+2\cdot Y\cdot X+Y^2-(\alpha-\beta)\cdot(\alpha+\beta)$

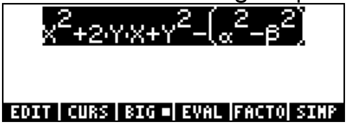
Press ↱ UNDO to recover the original expression. Now, let's select the entire expression by pressing the upper arrow key (▲) once. And press the FACTO soft menu key, to get



$\left(X+Y+\sqrt{\alpha^2-\beta^2}\right)\cdot\left(X+Y-\sqrt{\alpha^2-\beta^2}\right.$

Press ↱ UNDO to recover the original expression.

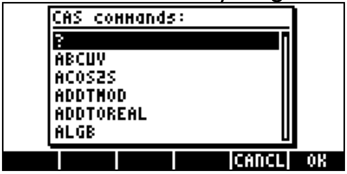
**Note**: Pressing the EVAL or the SIMP soft menu keys, while the entire original expression is selected, produces the following simplification of the expression:



$X^2+2\cdot Y\cdot X+Y^2-\left(\alpha^2-\beta^2\right)$

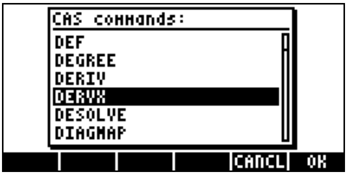
## Using the CMDS menu key

With the original polynomial expression used in the previous exercise still selected, press the NXT key to show the CMDS and HELP soft menu keys. These two commands belong to the second part of the soft menu available with the Equation Writer. Let's try this example as an application of the CMDS soft menu key: Press the CMDS soft menu key to get the list of CAS commands:



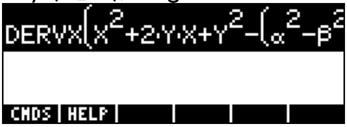
CAS commands:
?
ABCUV
ACOS2S
ADDTMOD
ADDTOREAL
ALGB

Next, select the command DERVX (the derivative with respect to the variable X, the current CAS independent variable) by using: ALPHA D ▼ ▼ ▼ . Command DERVX will now be selected:



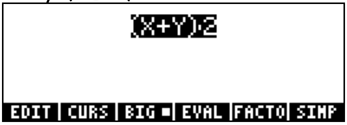
CAS commands:
DEF
DEGREE
DERIV
DERVX
DESOLVE
DIAGMAP

Press the OK soft menu key (F6), to get:



$\mathrm{DERVX}\left(X^2+2\cdot Y\cdot X+Y^2-\left(\alpha^2-\beta^2\right.\right.$

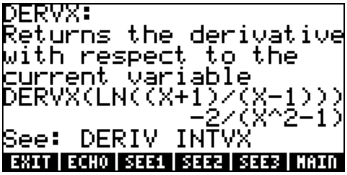
Next, press the NXT key to recover the original Equation Writer menu, and press the EVAL soft menu key (F4) to evaluate this derivative. The result is:



```
(X+Y)·2
```

### Using the HELP menu

Press the NXT key to show the CMDS and HELP soft menu keys. Press the HELP soft menu key to get the list of CAS commands. Then, press ALPHA D ▼ ▼ ▼ to select the command DERVX. Press the OK soft menu key (F6), to get information on the command DERVX:



```
DERVX:
Returns the derivative
with respect to the
current variable
DERVX(LN((X+1)/(X-1)))
             -2/(X^2-1)
See: DERIV INTVX
```

Detailed explanation on the use of the help facility for the CAS is presented in Chapter 1. To return to the Equation Writer, press the EXIT soft menu key. Press the ENTER key to exit the Equation Writer.

### Using the editing functions BEGIN, END, COPY, CUT and PASTE

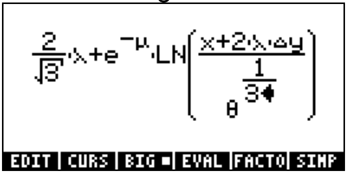
To facilitate editing, whether with the Equation Writer or on the stack, the calculator provides five editing functions, BEGIN, END, COPY, CUT and PASTE, activated by combining the right-shift key (↱) with keys (2,1), (2,2), (3,1), (3,2), and (3,3), respectively. These keys are located in the leftmost part of rows 2 and 3. The action of these editing functions are as follows:

BEGIN: marks the beginning of a string of characters for editing
END: marks the ending of a string of characters for editing
COPY: copies the string of characters selected by BEGIN and END
CUT: cuts the string of characters selected by BEGIN and END
PASTE: pastes a string of characters, previously copied or cut, into the current cursor position

To see and example, lets start the Equation Writer and enter the following expression (used in an earlier exercise):

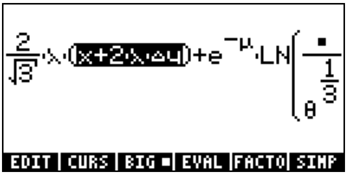
2 ÷ √x 3 ▶ ▶ × ALPHA ↱ M + ↰ eˣ +/- ALPHA ↱ M
▶ ▶ × ↱ LN ALPHA ↰ X + 2 × ALPHA ↱ M × ALPHA ↱ C
ALPHA ↰ Y ▲ ▲ ▲ ÷ ALPHA ↱ T yˣ 1 ÷ 3

The original expression is the following:



We want to remove the sub-expression $x+2\cdot\lambda\cdot\Delta y$ from the argument of the *LN* function, and move it to the right of the $\lambda$ in the first term. Here is one possibility: ▼ ◀ ◀ ◀ ▲ ▲ ▲ ↱ CUT ◀ ◀ ▲ × ↱ PASTE

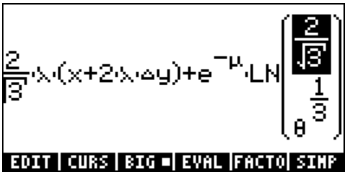
The modified expression looks as follows:



Next, we'll copy the fraction $2/\sqrt{3}$ from the leftmost factor in the expression, and place it in the numerator of the argument for the LN function. Try the following keystrokes:

▼ ▼ ◀ ◀ ▲ ▲ ▲ ↱ COPY ▼ ▼
↱ ▶ ◀ ◀ ◀ ↱ PASTE

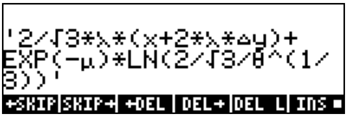
The resulting screen is as follows:



The functions BEGIN and END are not necessary when operating in the Equation Writer, since we can select strings of characters by using the arrow keys. Functions BEGIN and END are more useful when editing an expression with the line editor. For example, let's select the expression $x+2\cdot\lambda\cdot\Delta y$ from this expression, but using the line editor within the Equation Writer, as follows:

↱ ▲ F1

The line editor screen will look like this (quotes shown only if calculator in RPN mode):

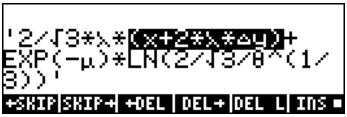


```
'2/√3*λ*(x+2*λ*∆y)+
EXP(-μ)*LN(2/√3/θ^(1/
3))'
```

To select the sub-expression of interest, use:

▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ↱ BEGIN
▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ▶ ↱ END

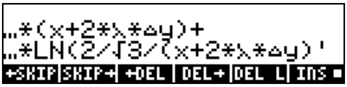
The screen shows the required sub-expression highlighted:



```
'2/√3*λ*(x+2*λ*∆y)+
EXP(-μ)*LN(2/√3/θ^(1/
3))'
```

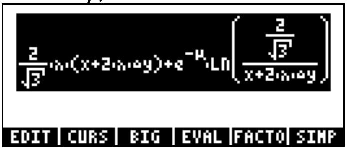
We can now copy this expression and place it in the denominator of the LN argument, as follows: ↱ COPY ▶ ▶ … (27 times) … ▶
◀ ◀ … (9 times) … ◀ ↱ PASTE

The line editor now looks like this:



```
…*(x+2*λ*∆y)+
…*LN(2/√3/(x+2*λ*∆y)'
```

Pressing ENTER shows the expression in the Equation Writer (in small-font format, press the BIG F3 soft menu key):



$$\frac{2}{\sqrt{3}}\cdot\lambda\cdot(x+2\cdot\lambda\cdot\Delta y)+e^{-\mu}\cdot LN\left(\frac{\frac{2}{\sqrt{3}}}{x+2\cdot\lambda\cdot\Delta y}\right)$$

Press ENTER to exit the Equation Writer.

## Creating and editing summations, derivatives, and integrals

Summations, derivatives, and integrals are commonly used for calculus, probability and statistics applications. In this section we show some examples of such operations created with the equation writer. Use ALG mode.

## Summations

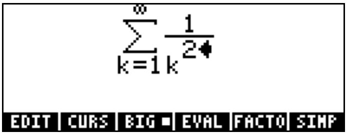
We will use the Equation Writer to enter the following summation:

$$\sum_{k=1}^{\infty}\frac{1}{k^2}$$

Press ↱ EQW to activate the Equation Writer. Then press ↱ Σ to enter the summation sign. Notice that the sign, when entered into the Equation Writer screen, provides input locations for the index of the summation as well as for the quantity being summed. To fill these input locations, use the following keystrokes:

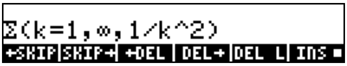
ALPHA ↰ K ▶ 1 ▶ ↰ ∞ ▶ 1 ÷ ALPHA ↰ K $y^x$ 2

The resulting screen is:



$$\sum_{k=1}^{\infty} \frac{1}{k^{2}}$$

To see the corresponding expression in the line editor, press ↱ ▲ and the F1 soft menu key, to show:

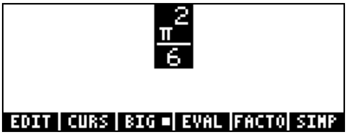


```
Σ(k=1,∞,1/k^2)
```

This expression shows the general form of a summation typed directly in the stack or line editor:

Σ(*index = starting_value, ending_value, summation expression*)

Press ENTER to return to the Equation Writer. The resulting screen shows the value of the summation,



$$\frac{\pi^{2}}{6}$$

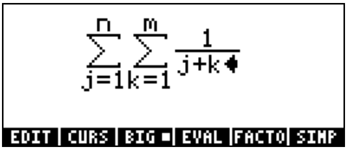
To recover the unevaluated summation use ↱ UNDO. To evaluate the summation again, you can use the F4 soft menu key. This shows again that

$$\sum_{k=1}^{\infty} \frac{1}{k^{2}} = \frac{\pi^{2}}{6}.$$

You can use the Equation Writer to prove that

$$\sum_{k=1}^{\infty} \frac{1}{k} = +\infty .$$

This summation (representing an infinite series) is said to diverge.
Double summations are also possible, for example:


$$\sum_{j=1}^{n}\sum_{k=1}^{m}\frac{1}{j+k}$$

### Derivatives

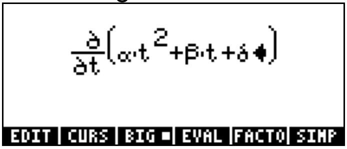
We will use the Equation Writer to enter the following derivative:

$$\frac{d}{dt}(\alpha\cdot t^2+\beta\cdot t+\delta)$$

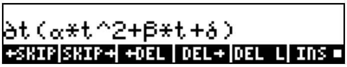
Press ↱ EQW to activate the Equation Writer. Then press ↱ ∂ to enter the (partial) derivative sign. Notice that the sign, when entered into the Equation Writer screen, provides input locations for the expression being differentiated and the variable of differentiation. To fill these input locations, use the following keystrokes:

ALPHA ↰ T ▶ ALPHA ↱ A × ALPHA ↰ T $Y^x$ 2
▶ ▶ + ALPHA ↱ B × ALPHA ↰ T + ALPHA ↱ D

The resulting screen is the following:

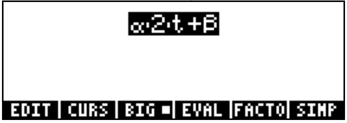

$$\frac{\partial}{\partial t}\left(\alpha\cdot t^{2}+\beta\cdot t+\delta\right)$$

To see the corresponding expression in the line editor, press ↱ ▲ and the F1 soft menu key, to show:


```
∂t(α*t^2+β*t+δ)
```

This indicates that the general expression for a derivative in the line editor or in the stack is: ∂*variable*(*function of variables*)

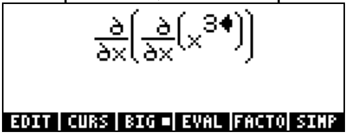
Press ENTER to return to the Equation Writer. The resulting screen is not the derivative we entered, however, but its symbolic value, namely,


$\alpha\cdot 2\cdot t+\beta$

To recover the derivative expression use ↱ UNDO . To evaluate the derivative again, you can use the F4 soft menu key. This shows again that

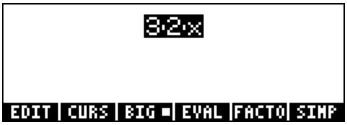
$$\frac{d}{dt}(\alpha \cdot t^2 - \beta \cdot t + \delta) = 2\alpha \cdot t + \beta \ .$$

Second order derivatives are possible, for example:



$\frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\left(x^{3}\right)\right)$

which evaluates to:



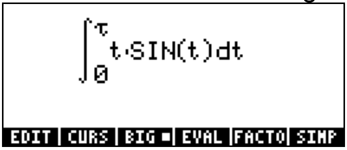
3·2·x

**Note**: The notation $\frac{\partial}{\partial x}(\ )$ is proper of partial derivatives. The proper notation for total derivatives (i.e., derivatives of one variable) is $\frac{d}{dx}(\ )$. The calculator, however, does not distinguish between partial and total derivatives.

### Definite integrals

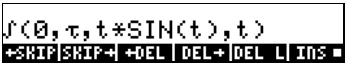
We will use the Equation Writer to enter the following definite integral: $\int_0^{\tau} t \cdot \sin(t) \cdot dt$ . Press ↱ EQW to activate the Equation Writer.

Then press ↱ ∫ to enter the integral sign. Notice that the sign, when entered into the Equation Writer screen, provides input locations for the limits of integration, the integrand, and the variable of integration. To fill these input locations, use the following keystrokes: 0 ▶ ALPHA ↱ U ▶ ALPHA ↰ T × SIN ALPHA ↰ T ▶ ALPHA ↰ T . The resulting screen is the following:



$\int_0^{\tau} t \cdot SIN(t) dt$

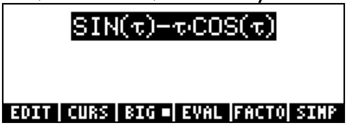
To see the corresponding expression in the line editor, press ▲ ▲ and the F1 soft menu key, to show:



∫(0,τ,t*SIN(t),t)

This indicates that the general expression for a derivative in the line editor or in the stack is: ∫(*lower_limit, upper_limit,integrand,variable_of_integration*)

Press ENTER to return to the Equation Writer. The resulting screen is not the definite integral we entered, however, but its symbolic value, namely,

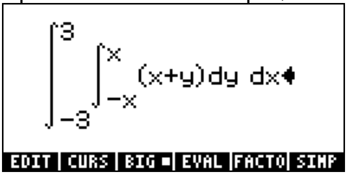


SIN(τ)−τ·COS(τ)

To recover the derivative expression use ↱ UNDO . To evaluate the derivative again, you can use the F4 soft menu key. This shows again that

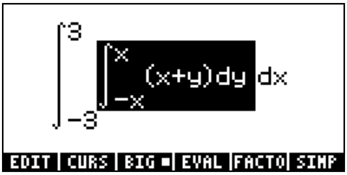
$$\int_0^{\tau} t \cdot \sin(t) \cdot dt = \sin(\tau) - \tau \cdot \cos(\tau)$$

Double integrals are also possible. For example,

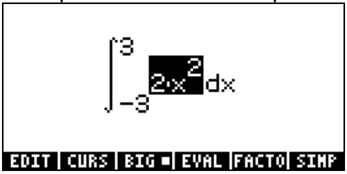


$$\int_{-3}^{3}\int_{-x}^{x}(x+y)dy\ dx$$

which evaluates to 36. Partial evaluation is possible, for example:



$$\int_{-3}^{3}\int_{-x}^{x}(x+y)dy\ dx$$

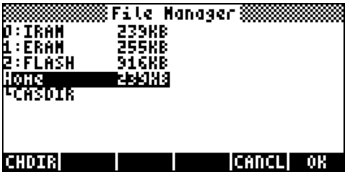


$$\int_{-3}^{3}2\cdot x^2 dx$$

This integral evaluates to 36.

## Organizing data in the calculator

You can organize data in your calculator by storing variables in a directory tree. To understand the calculator's memory, we first take a look at the file directory. Press the keystroke combination ↰ FILES (first key in second row of keys from the top of the keyboard) to get the calculator's File Manager screen:



```
File Manager
0:IRAM     239KB
1:ERAM     255KB
2:FLASH    916KB
Home       239KB
└CASDIR
CHDIR          CANCL  OK
```

This screen gives a snapshot of the calculator's memory and of the directory tree. The screen shows that the calculator has three memory ports (or memory partitions), port *0:IRAM*, *port 1:ERAM*, and *port 2:FLASH* . Memory ports are used to store third party application or libraries, as well as for backups. The size of the three different ports is also indicated. The fourth and subsequent lines in this screen show the calculator's directory tree. The top directory (currently highlighted) is the *Home* directory, and it has predefined into it a sub-directory called *CASDIR*. The *File Manager* screen has three functions associated with the soft-menu keys:

CHDIR (F1): Change to selected directory
CANCL (F5): Cancel action
OK (F6): Approve a selection

For example, to change directory to the CASDIR, press the down-arrow key, ▼, and press CHDIR (F1). This action closes the *File Manager* window and returns us to normal calculator display. You will notice that the second line from the top in the display now starts with the characters { HOME CASDIR } indicating that the current directory is CASDIR within the HOME directory.

## Functions for manipulation of variables

This screen includes 20 commands associated with the soft menu keys that can be used to create, edit, and manipulate variables. The first six functions are the following:

EDIT To edit a highlighted variable
COPY To copy a highlighted variable
MOVE To move a highlighted variable
RCL To recall the contents of a highlighted variable
EVAL To evaluate a highlighted variable
TREE To see the directory tree where the variable is contained

If you press the NXT key, the next set of functions is made available:

PURGE To purge, or delete, a variable

RENAM To rename a variable
NEW To create a new variable
ORDER To order a set of variables in the directory
SEND To send a variable to another calculator or computer
RECV To receive a variable from another calculator or computer

If you press the NXT key, the third set of functions is made available:

HALT To return to the stack temporarily
VIEW To see contents of a variable
EDITB To edit contents of a binary variable (similar to EDIT)
HEADE To show the directory containing the variable in the header
LIST Provides a list of variable names and description
SORT To sort variables according to a sorting criteria

If you press the NXT key, the last set of functions is made available:

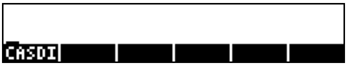
XSEND To send variable with X-modem protocol
CHDIR To change directory

To move between the different soft menu commands, you can use not only the NEXT key (NXT), but also the PREV key (↰ PREV).

The user is invited to try these functions on his or her own. Their applications are straightforward.

## The HOME directory

The HOME directory, as pointed out earlier, is the base directory for memory operation for the calculator. To get to the HOME directory, you can press the UPDIR function (↰ UPDIR) -- repeat as needed -- until the { HOME } spec is shown in the second line of the display header. Alternatively, you can use ↰(hold) UPDIR, press ENTER if in the algebraic mode. For this example, the HOME directory contains nothing but the CASDIR. Pressing VAR will show the variables in the soft menu keys:

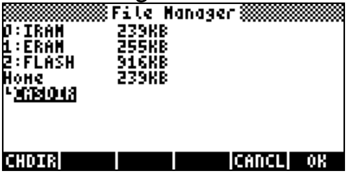


### Subdirectories

To store your data in a well organized directory tree you may want to create subdirectories under the HOME directory, and more subdirectories within subdirectories, in a hierarchy of directories similar to folders in modern computers. The subdirectories will be given names that may reflect the contents of each subdirectory, or any arbitrary name that you can think of.

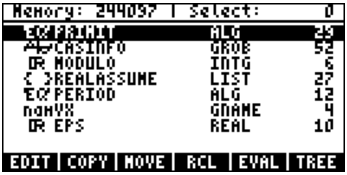
## The CASDIR sub-directory

The CASDIR sub-directory contains a number of variables needed by the proper operation of the CAS (Computer Algebraic System, see appendix C). To see the contents of the directory, we can use the keystroke combination: ↰ FILES which opens the *File Manager* once more:



```
       File Manager
0:IRAM       239KB
1:ERAM       255KB
2:FLASH      916KB
Home         239KB
└CASDIR
CHDIR |     |     |     |CANCL| OK
```

This time the CASDIR is highlighted in the screen. To see the contents of the directory press the OK (F6) soft menu key or ENTER, to get the following screen:



```
Memory: 244097 | Select:      0
 EQ PRIMIT            ALG     29
 ▀▄ CASINFO           GROB    52
 |R MODULO            INTG     6
 { }REALASSUME        LIST    27
 EQ PERIOD            ALG     12
 namVX                GNAME    4
 |R EPS               REAL    10
EDIT |COPY |MOVE | RCL |EVAL |TREE
```

The screen shows a table describing the variables contained in the CASDIR directory. These are variables pre-defined in the calculator memory that establish certain parameters for CAS operation (see appendix C). The table above contains 4 columns:

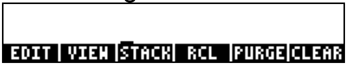
- The first column indicate the type of variable (e.g., 'EQ' means an equation-type variable, |R indicates a real-value variable, { } means a list, *nam* means 'a global name', and the symbol ▀▄ represents a graphic variable.
- The second column represents the name of the variables, i.e., *PRIMIT, CASINFO, MODULO, REALASSUME, PERIOD, VX,* and *EPS*.
- Column number 3 shows another specification for the variable type, e.g., *ALG* means an algebraic expression, *GROB* stands for *graphics object, INTG* means an integer numeric variable, *LIST* means a list of data,

GNAME means a *global name*, and *REAL* means a real (or floating-point) numeric variable.

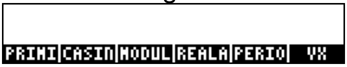
- The fourth and last column represents the size, in bytes, of the variable truncated, without decimals (i.e., nibbles). Thus, for example, variable PERIOD takes 12.5 bytes, while variable REALASSUME takes 27.5 bytes (1 byte = 8 bits, 1 bit is the smallest unit of memory in computers and calculators).

## CASDIR Variables in the stack

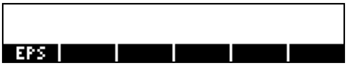
Pressing the ON key closes the previous screen and returns us to normal calculator display. By default, we get back the TOOL menu:



We can see the variables contained in the current directory, CASDIR, by pressing the VAR key (first key in the second row from the top of the keyboard). This produces the following screen:



Pressing the NXT key shows one more variable stored in this directory:



- To see the contents of the variable EPS, for example, use ↱ EPS. This shows the value of EPS to be .0000000001
- To see the value of a numerical variable, we need to press only the soft menu key for the variable. For example, pressing EPS followed by ENTER, shows the same value of the variable in the stack, if the calculator is set to *Algebraic*. If the calculator is set to *RPN* mode, you need only press the soft menu key for ENTER.
- To see the full name of a variable, press the apostrophe first ', and then the soft menu key corresponding to the variable. For example, for the variable listed in the stack as PERIO, we use: ' PERIO, which produces as output the string: 'PERIOD'. This procedure applies to both the *Algebraic* and *RPN* calculator operating modes.

## Variables in CASDIR

The default variables contained in the CASDIR directory are the following:

*PRIMIT* Latest primitive (anti-derivative) calculated, not a default

| | |
|---|---|
| | variable, but one created by a previous exercise |
| *CASINFO* | a graph that provides CAS information |
| *MODULO* | Modulo for modular arithmetic (default = 13) |
| *REALASSUME* | List of variable names assumed as real values |
| *PERIOD* | Period for trigonometric functions (default = $2\pi$) |
| *VX* | Name of default independent variable (default = X) |
| *EPS* | Value of small increment (epsilon), (default = $10^{-10}$) |

These variables are used for the operation of the CAS.

## Typing directory and variable names

To name subdirectories, and sometimes, variables, you will have to type strings of letters at once, which may or may not be combined with numbers. Rather than pressing the ALPHA, ALPHA ↰, or ALPHA ↱ key combinations to type each letter, you can hold down the ALPHA key and enter the various letter. You can also lock the alphabetic keyboard temporarily and enter a full name before unlocking it again. The following combinations of keystrokes will lock the alphabetic keyboard:

ALPHA ALPHA locks the alphabetic keyboard in upper case. When locked in this fashion, pressing the ↰ before a letter key produces a lower case letter, while pressing the ↱ key before a letter key produces a special character. If the alphabetic keyboard is already locked in upper case, to lock it in lower case, type, ↰ ALPHA

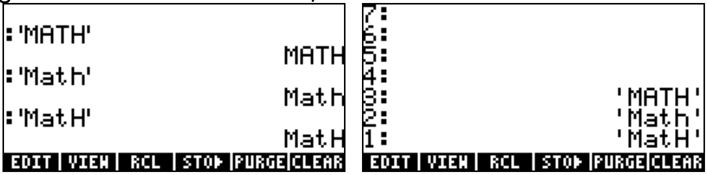
ALPHA ALPHA ↰ ALPHA locks the alphabetic keyboard in lower case. When locked in this fashion, pressing the ↰ before a letter key produces an upper case letter. To unlock lower case, press ↰ ALPHA

To unlock the upper-case locked keyboard, press ALPHA

Let's try some exercises typing directory/variable names in the stack. Assuming that the calculator is in the Algebraic mode of operation (although the instructions work as well in RPN mode), try the following keystroke sequences. With these commands we will be typing the words 'MATH', 'Math', and 'MatH'

' ALPHA ALPHA M A T H / ALPHA ENTER
' ALPHA ALPHA M ← A ← T ← H ALPHA ENTER
' ALPHA ALPHA M ← ALPHA A T ← H ALPHA ENTER

The calculator display will show the following (left-hand side is Algebraic mode, right-hand side is RPN mode):



```
:'MATH'
                    MATH
:'Math'
                    Math
:'MatH'
                    MatH
```

```
7:
6:
5:
4:
3:                'MATH'
2:                'Math'
1:                'MatH'
```

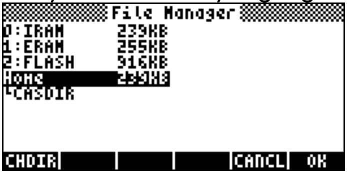
**Note:** if system flag 60 is set, you can lock the alphabetical keyboard by just pressing ALPHA. See Chapter 1 for more information on system flags.

## Creating subdirectories

Subdirectories can be created by using the FILES environment or by using the command CRDIR. The two approaches for creating sub-directories are presented next.

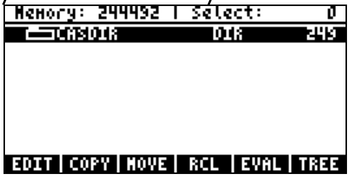
### Using the FILES menu

Regardless of the mode of operation of the calculator (Algebraic or RPN), we can create a directory tree, based on the HOME directory, by using the functions activated in the FILES menu. Press ← FILES to activate the FILES menu. If the HOME directory is not already highlighted in the screen, i.e.,



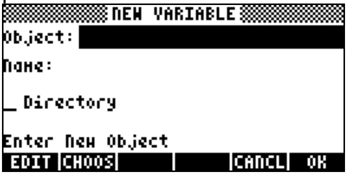
```
File Manager
0:IRAM      239KB
1:ERAM      255KB
2:FLASH     916KB
HOME        244KB
└CASDIR
```

use the up and down arrow keys (▲ ▼) to highlight it. Then, press the OK (F6) soft menu key. The screen may look like this:



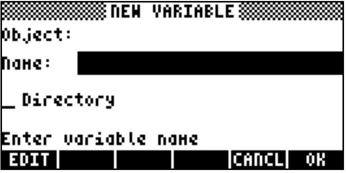
```
Memory: 244492 | Select:     0
  CASDIR          DIR       249
```

showing that only one object exists currently in the HOME directory, namely, the CASDIR sub-directory.  Let's create another sub-directory called MANS (for MANualS) where we will store variables developed as exercises in this manual.  To create this sub-directory first enter: NXT NEW (F3) . This will produce the following input form:



```
NEW VARIABLE
Object:
Name:
_ Directory
Enter New Object
```

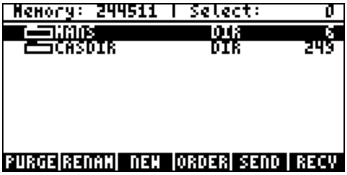
The *Object* input field, the first input field in the form, is highlighted by default. This input field can hold the contents of a new variable that is being created. Since we have no contents for the new sub-directory at this point, we simply skip this input field by pressing the down-arrow key, ▼, once.  The *Name* input field is now highlighted:



```
NEW VARIABLE
Object:
Name:
_ Directory
Enter variable name
```

This is where we enter the name of the new sub-directory (or variable, as the case may be), as follows: ALPHA ALPHA M A N S ENTER

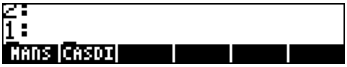
The cursor moves to the *_Directory* check field.  Press the CHK (F3) soft menu key to specify that you are creating a directory, and press OK to exit the input form.  The variable listing for the HOME directory will be shown in the screen as follows:



```
Memory: 244511 | Select:      0
MANS              DIR         6
CASDIR            DIR       249
```

The screen indicates that there is a new directory (MANS) within the HOME directory.

Next, we will create a sub-directory named INTRO (for INTROduction), within MANS, to hold variables created as exercise in subsequent sections of this chapter. Press the ON key to return to normal calculator display (the TOOLS menu will be shown). Then, press VAR to show the HOME directory contents in the soft menu key labels. The display may look like this (if you have created other variables in the HOME directory they will show in the soft menu key labels too):

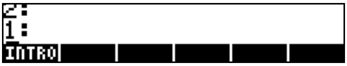


To move into the MANS directory, press the corresponding soft menu key (F1 in this case), and ENTER if in algebraic mode. The directory tree will be shown in the second line of the display as { HOME MANS }. However, there will be no labels associated with the soft menu keys, as shown below, because there are no variables defined within this directory.

Let's create the sub-directory INTRO by using:

↰ FILES OK NXT NEW ▼ ALPHA ALPHA I N T R O ENTER CHK OK

Press the ON key, followed by the VAR key, to see the contents of the MANS directory as follows:

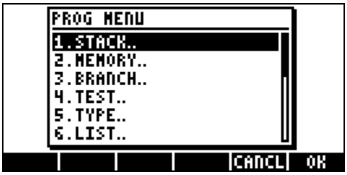


Press the INTRO soft menu key to move into the INTRO sub-directory. This will show an empty sub-directory. Later on, we will do some exercises in creating variables.

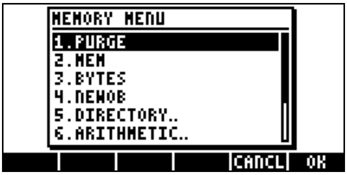
### Using the command CRDIR

The command CRDIR can be used to create directories. This command is available through the command catalog key (the ↱ CAT key, second key in fourth row of keys from the top of the keyboard), through the programming menus (the ↰ PRG key, same key as the ↱ CAT key), or by simply typing it.

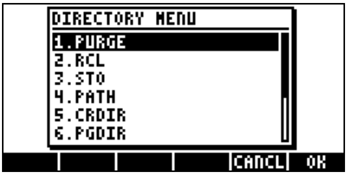
- Through the catalog key
  Press ↱ CAT ALPHA C . Use the up and down arrow keys (▲▼) to locate the CRDIR command. Press the OK soft menu key to activate the command.
- Through the programming menus
  Press ↰ PRG . This will produce the following pull-down menu for programming:



Use the down arrow key (▼) to select the option *2. MEMORY…* , or just press 2. Then, press OK. This will produce the following pull-down menu:



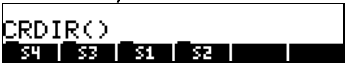
Use the down arrow key (▼) to select the *5. DIRECTORY* option, or just press 5. Then, press OK. This will produce the following pull-down menu:



Use the down arrow key (▼) to select the *5. CRDIR* option, and press OK.

### *Command CRDIR in Algebraic mode*

Once you have selected the CRDIR through one of the means shown above, the command will be available in your stack as follows:

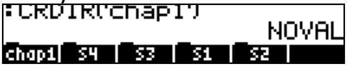


```
CRDIR()
```

At this point, you need to type a directory name, say *chap1* :

ALPHA ALPHA ← ALPHA C H A P 1 ALPHA ENTER

The name of the new directory will be shown in the soft menu keys, e.g.,



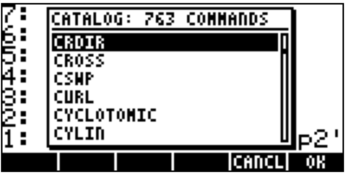
```
:CRDIR('chap1')
                 NOVAL
```

### *Command CRDIR in RPN mode*

To use the CRDIR in RPN mode you need to have the name of the directory already available in the stack before accessing the command. For example:

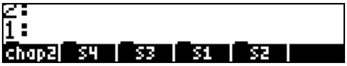
ALPHA ALPHA ← ALPHA C H A P 2 ALPHA ENTER

Then access the CRDIR command by either of the means shown above, e.g., through the → CAT key:



```
7:  CATALOG: 763 COMMANDS
6:  CRDIR
5:  CROSS
4:  CSWP
3:  CURL
2:  CYCLOTOMIC
1:  CYLIN                   p2'
                  CANCL  OK
```

Press the OK soft menu key to activate the command, to create the sub-directory:



```
2:
1:
chap2  S4  S3  S1  S2
```

## Moving among subdirectories

To move down the directory tree, you need to press the soft menu key corresponding to the sub-directory you want to move to. The list of variables in a sub-directory can be produced by pressing the VAR (VARiables) key. To move up in the directory tree, use the function UPDIR, i.e., enter ← UPDIR .

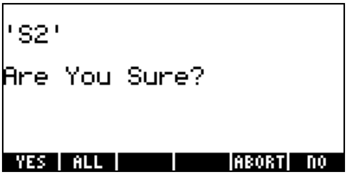
Alternatively, you can use the FILES menu, i.e., press ← FILES . Use the up and down arrow keys (▲ ▼) to select the sub-directory you want to move to, and then press the CHDIR (CHange DIRectory) or F1 soft menu key. This will show the contents of the sub-directory you moved to in the soft menu key labels.

## Deleting subdirectories

To delete a sub-directory, use one of the following procedures:

### Using the FILES menu

Press the ← FILES key to trigger the FILES menu. Select the directory containing the sub-directory you want to delete, and press the CHDIR if needed. This will close the FILES menu and display the contents of the directory you selected. In this case you will need to press ENTER. Press the OK soft menu key to list the contents of the directory in the screen. Select the sub-directory (or variable) that you want to delete. Press NXT PURGE. A screen similar to the following will be shown:

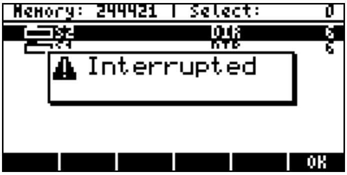

'S2'

Are You Sure?

The 'S2' string in this form is the name of the sub-directory that is being deleted. The soft menu keys provide the following options:

YES (F1) Proceed with deleting the sub-directory (or variable)
ALL (F2) Proceed with deleting all sub-directories (or variables)
ABORT (F5) Do not delete sub-directory (or variable) from a list
NO (F6) Do not delete sub-directory (or variable)

After selecting one of these four commands, you will be returned to the screen listing the contents of the sub-directory. The ABORT command, however, will show an error message:


Memory: 244421 | Select: 0

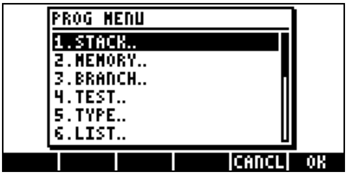
Interrupted

and you will have to press OK, before returning to the variable listing.

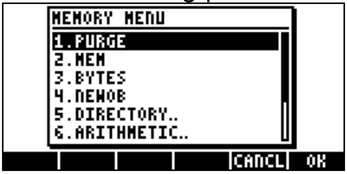
### Using the command PGDIR

The command PGDIR can be used to purge directories. Like the command CRDIR, the PGDIR command is available through the ↱ CAT or through the ↰ PRG key, or it can simply be typed in.

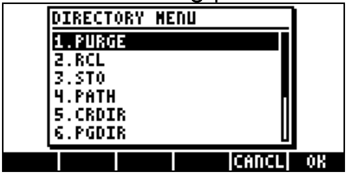
- Through the catalog key
  Press ↱ CAT ALPHA ALPHA P G. This should highlight the PGDIR command. Press the OK soft menu key to activate the command.
- Through the programming menus
  Press ↰ PRG. This will produce the following pull-down menu for programming:



Use the down arrow key (▼) to select the option *2. MEMORY…* Then, press OK. This will produce the following pull-down menu:



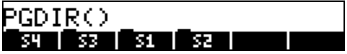
Use the down arrow key (▼) to select the *5. DIRECTORY* option. Then, press OK. This will produce the following pull-down menu:



Use the down arrow key (▼) to select the *6. PGDIR* option, and press OK.

### *Command PGDIR in Algebraic mode*

Once you have selected the PGDIR through one of the means shown above, the command will be available in your stack as follows:

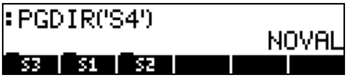


```
PGDIR()
```

At this point, you need to type the name of an existing directory, say *S4* :

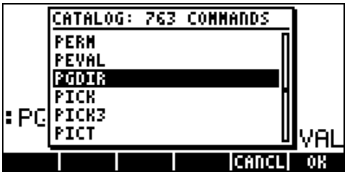
ALPHA S 4 ENTER

As a result, sub-directory S4 is deleted:



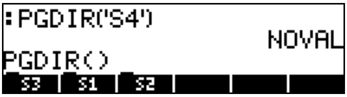
```
:PGDIR('S4')
                NOVAL
```

Instead of typing the name of the directory, you can simply press the corresponding soft menu key at the listing of the PGDIR( ) command, e.g.,



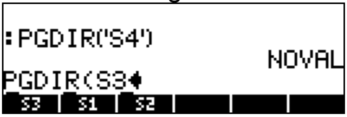
```
:PG                                   VAL
```

Press OK, to get:



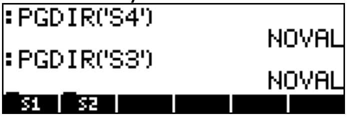
```
:PGDIR('S4')
                                  NOVAL
PGDIR()
```

Then, press S3 to enter 'S3' as the argument to PGDIR.



```
:PGDIR('S4')
                                  NOVAL
PGDIR(S3◆
```

Press ENTER to delete the sub-directory:

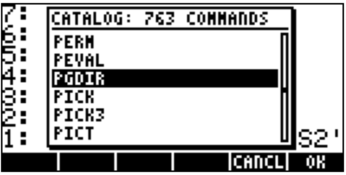


```
:PGDIR('S4')
                                  NOVAL
:PGDIR('S3')
                                  NOVAL
```

### *Command PGDIR in RPN mode*

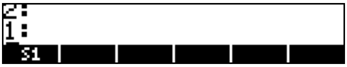
To use the PGDIR in RPN mode you need to have the name of the directory, between quotes, already available in the stack before accessing the command. For example: ' ALPHA S 2 ENTER

Then access the PGDIR command by either of the means shown above, e.g., through the → CAT key:



```
7:
6:
5:
4:
3:
2:
1:                                  S2'
```

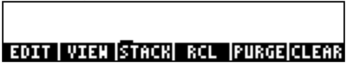
Press the OK soft menu key to activate the command and delete the sub-directory:

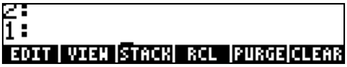


```
2:
1:
```

### Using the PURGE command from the TOOL menu

The TOOL menu is available by pressing the TOOL key (Algebraic and RPN modes shown):





The PURGE command is available by pressing the PURGE soft menu key (F5 ). In the following examples we want to delete sub-directory *S1*:

- Algebraic mode: Enter PURGE VAR S1 ENTER
- RPN mode: Enter VAR ' S1 ENTER TOOL PURGE VAR

# Variables

Variables are like files on a computer hard drive. One variable can store one object (numerical values, algebraic expressions, lists, vectors, matrices, programs, etc). Even sub-directories can be through of as variables (in fact, in the calculator, a subdirectory is also a type of calculator object).

Variables are referred to by their names, which can be any combination of alphabetic and numerical characters, starting with a letter (either English or Greek). Some non-alphabetic characters, such as the arrow (→) can be used in a variable name, if combined with an alphabetical character. Thus, '→A' is a valid variable name, but '→' is not. Valid examples of variable names are: 'A', 'B', 'a', 'b', 'α', 'β', 'A1', 'AB12', '→A12','Vel','Z0','z1', etc.

A variable can not have the same name than a function of the calculator. You can not have a SIN variable for example as there is a SIN command in the calculator. The reserved calculator variable names are the following: ALRMDAT, CST, EQ, EXPR, IERR, IOPAR, MAXR, MINR, PICT, PPAR, PRTPAR, VPAR, ZPAR, der_, e, i, n1,n2, …, s1, s2, …, ΣDAT, ΣPAR, π, ∞

Variables can be organized into sub-directories.

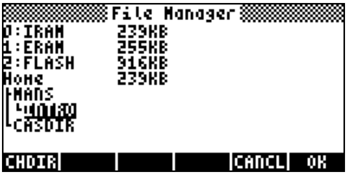
## Creating variables

To create a variable, we can use the FILES menu, along the lines of the examples shown above for creating a sub-directory. For example, within the sub-directory {HOME MANS INTRO}, created in an earlier example, we want to store the following variables with the values shown:

| ***Name*** | ***Contents*** | ***Type*** |
|---|---|---|
| A | 12.5 | real |
| $\alpha$ | -0.25 | real |
| A12 | $3\times10^5$ | real |
| Q | ‘r/(m+r)' | algebraic |
| R | [3,2,1] | vector |
| z1 | 3+5i | complex |
| p1 | << → r 'π*r^2' >> | program |

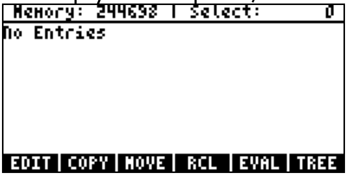
### Using the FILES menu

We will use the FILES menu to enter the variable A. We assume that we are in the sub-directory {HOME MANS INTRO}. To get to this sub-directory, use the following: ← FILES and select the INTRO sub-directory as shown in this screen:



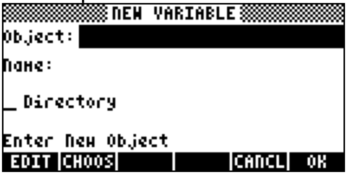
File Manager
0:IRAM 239KB
1:ERAM 255KB
2:FLASH 916KB
Home 239KB
MANS
INTRO
CASDIR

Press OK to enter the directory. You will get a files listing with no entries (the INTRO sub-directory is empty at this point)



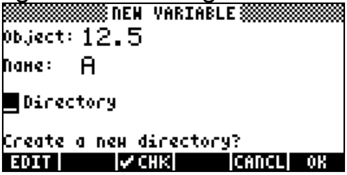
Memory: 244698 | Select: 0
No Entries

Press the NXT key to move to the next set of soft menu keys, and press the NEW soft menu key. This will produce the NEW VARIABLE input form:



NEW VARIABLE
Object:
Name:
_ Directory
Enter New Object

To enter variable A (see table above), we first enter its contents, namely, the number 12.5, and then its name, A, as follows: 1 2 . 5 OK ALPHA A OK. Resulting in the following screen:



NEW VARIABLE

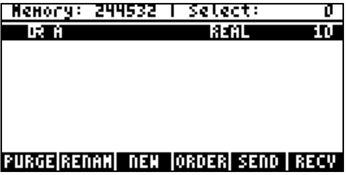
Object: 12.5

Name: A

□ Directory

Create a new directory?

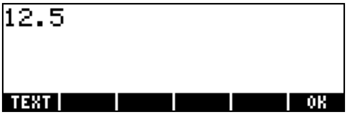
Press OK once more to create the variable. The new variable is shown in the following variable listing:



```
Memory: 244532 | Select:       0
  ℝ A                 REAL      10
```

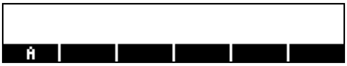
The listing indicates a real variable (|R), whose name is A, and that occupies 10.5 bytes of memory. To see the contents of the variable in this screen, press NXT VIEW.

- Press the GRAPH ( F1 ) soft menu key to see the contents in a graphical format.



12.5

- Press the TEXT ( F1 ) soft menu key to see the contents in text format.
- Press OK to return to the variable list
- Press ON once more to return to normal display. Variable A should now be featured in the soft menu key labels:



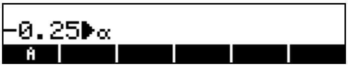
### Using the STO▶ command

A simpler way to create a variable is by using the STO command (i.e., the STO▶ key). We provide examples in both the Algebraic and RPN modes, by creating the remaining of the variables suggested above, namely:

| ***Name*** | ***Contents*** | ***Type*** |
|---|---|---|
| α | -0.25 | real |
| A12 | $3\times10^5$ | real |
| Q | ‘r/(m+r)' | algebraic |
| R | [3,2,1] | vector |
| z1 | 3+5i | complex |
| p1 | << → r 'π*r^2' >> | program |

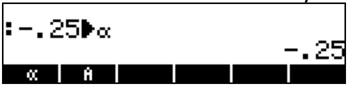
- ***Algebraic mode***

  Use the following keystrokes to store the value of –0.25 into variable α: 0 • 2 5 +/- STO▶ ALPHA ↱ A . AT this point, the screen will look as follows:



```
-0.25▶α
 A
```

This expression means that the value –0.25 is being stored into α (the symbol ▶ suggests the operation). Press ENTER to create the variable. The variable is now shown in the soft menu key labels:



```
:-.25▶α
                 -.25
 α    A
```

The following are the keystrokes required to enter the remaining variables:

A12: 3 EEX 5 STO▶ ALPHA A 1 2 ENTER

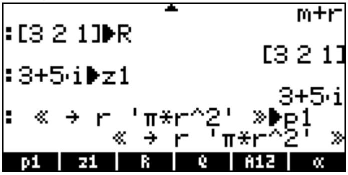
Q: ' ALPHA ↰ R ÷ ↰ () ALPHA ↰ M + ALPHA ↰ R ▶ ▶ STO▶ ALPHA Q ENTER

R: ↰ [] 3 ↱ , 2 ↱ , 1 ▶ STO▶ ALPHA R ENTER

z1: 3 + 5 × ↰ i STO▶ ALPHA ↰ Z 1 ENTER (if needed, accept change to Complex mode)

p1: ↱ « » ↱ → ALPHA ↰ R ' ↰ π × ALPHA ↰ R yˣ 2 ▶ ▶ ▶ STO▶ ALPHA ↰ P 1 ENTER ..

The screen, at this point, will look as follows:

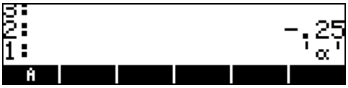


```
                          m+r
:[3 2 1]▶R
                      [3 2 1]
:3+5·i▶z1
                        3+5·i
: « → r 'π*r^2' »▶p1
      « → r 'π*r^2' »
 p1   z1   R    Q   A12   α
```

You will see six of the seven variables listed at the bottom of the screen: *p1, z1, R, Q, A12, α*.

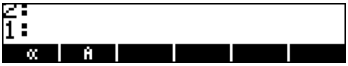
- ***RPN mode***

  Use the following keystrokes to store the value of –0.25 into variable α: 0 • 2 5 +/- ENTER ALPHA ↱ A ENTER. At this point, the screen will look as follows:



```
3:
2:                     -.25
1:                      'α'
 A
```

This expression means that the value –0.25 is ready to be stored into α. Press STO▶ to create the variable. The variable is now shown in the soft menu key labels:



```
2:
1:
 α  A
```

To enter the value 3×10$^5$ into A12, we can use a shorter version of the procedure: 3 EEX 5 ' ALPHA A 1 2 ENTER STO▶

Here is a way to enter the contents of Q:

Q: ' ALPHA ↰ R ÷ ↰ () ALPHA ↰ M + ALPHA ↰ R ▶ ▶ ' ALPHA Q ENTER STO▶

To enter the value of R, we can use an even shorter version of the procedure:

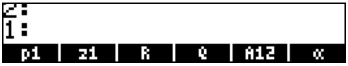
R: ↰ [] 3 SPC 2 SPC 1 ▶ ' ALPHA R ENTER STO▶

Notice that to separate the elements of a vector in RPN mode we can use the space key (SPC), rather than the comma (↱ , ) used above in Algebraic mode.

z1: ' 3 + 5 × ↰ i ' ALPHA ↰ Z 1 STO▶ (if needed, accept change to Complex mode)

p1: ↱ « » ↱ → ALPHA ↰ R ' ↰ π × ALPHA ↰ R yˣ 2 ▶ ▶ ▶ ' ALPHA ↰ P 1 ▶ ENTER STO▶.

The screen, at this point, will look as follows:



```
2:
1:
 p1  z1  R  Q  A12  α
```

You will see six of the seven variables listed at the bottom of the screen: *p1, z1, R, Q, A12, α*.

## Checking variables contents

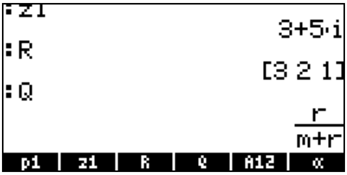
As an exercise on peeking into the contents of variables we will use the seven variables entered in the exercise above. We showed how to use the FILES menu to view the contents of a variable in an earlier exercise when we created the variable A. In this section we will show a simple way to look into the contents of a variable.

### Pressing the soft menu key label for the variable

This procedure will show the contents of a variable as long as the variable contains a numerical value or an algebraic value, or an array. For example, for the variables listed above, press the following keys to see the contents of the variables:

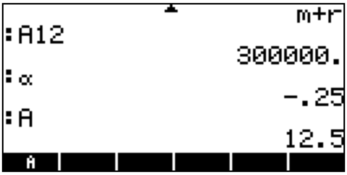
#### *Algebraic mode*

Type these keystrokes: VAR z1 ENTER R ENTER Q ENTER. At this point, the screen looks as follows:



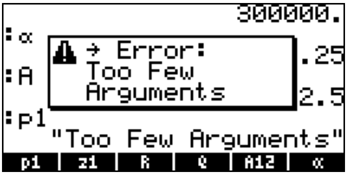
```
:z1
                 3+5·i
:R
               [3 2 1]
:Q
                    r
                   ───
                   m+r
 p1 │ z1 │ R │ Q │ A12 │ α
```

Next, type these keystrokes: A12 ENTER α ENTER NXT A ENTER. At this point, the screen looks as follows:



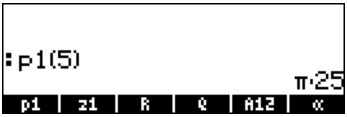
```
                   m+r
:A12
               300000.
:α
                  -.25
:A
                  12.5
 A │ │ │ │ │
```

Pressing the soft menu key corresponding to *p1* will provide an error message (try NXT p1 ENTER):



```
               300000.
:α
    → Error:
    Too Few
    Arguments
:A
:p1
  "Too Few Arguments"
 p1 │ z1 │ R │ Q │ A12 │ α
```

**Note:** By pressing p1 ENTER we are trying to activate (run) the *p1* program. However, this program expects a numerical input. Try the following exercise: ON p1 ← () 5 ENTER. The result is:



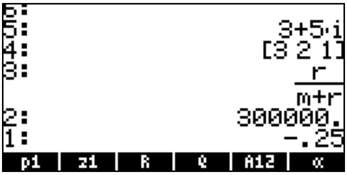
```
:p1(5)
                π·25
 p1 | z1 | R | Q | A12 | α
```

The program has the following structure: << → r 'π*r^2' >>
The « » symbols indicate a program in User RPL language (the original programming language of the HP 28/48 calculators, and available in the HP 49G series). The characters → r indicate that an input, to be referred to as *r*, is to be provided to the program. The action from the program is to take that value of *r* and evaluate the algebraic 'π*r^2'. In the example shown above, *r* took the value of 5, and thus the value of $\pi r^2 = \pi\cdot 25$ is returned. This program, therefore, calculates the area of a circle given its radius *r*.

### *RPN mode*

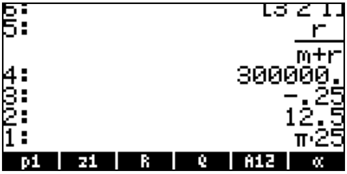
In RPN mode, you only need to press the corresponding soft menu key label to get the contents of a numerical or algebraic variable. For the case under consideration, we can try peeking into the variables *z1*, *R*, *Q*, *A12*, α, and *A*, created above, as follows: VAR z1 R Q A12 α

At this point, the screen looks like this:



```
6:
5:                 3+5·i
4:               [3 2 1]
3:                     r
                     ───
                     m+r
2:               300000.
1:                  -.25
 p1 | z1 | R | Q | A12 | α
```

To see the contents of A, use: NXT A
To run program *p1* with r = 5, use: NXT 5 p1.



```
6:               [3 2 1]
5:                     r
                     ───
                     m+r
4:               300000.
3:                  -.25
2:                  12.5
1:                  π·25
 p1 | z1 | R | Q | A12 | α
```

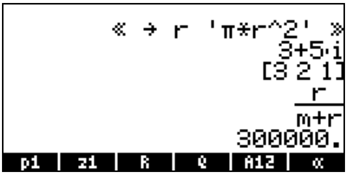
Notice that to run the program in RPN mode, you only need to enter the input (5) and press the corresponding soft menu key. (In algebraic mode, you need to place parentheses to enter the argument).

### Using the right-shift key ↱ followed by soft menu key labels

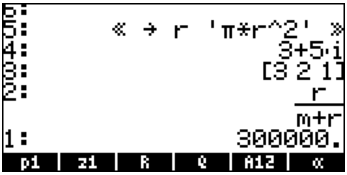
This approach for viewing the contents of a variable works the same in both Algebraic and RPN modes. Try the following examples in either mode:

VAR ↱ p1 ↱ z1 ↱ R ↱ Q ↱ A12

This produces the following screen (Algebraic mode in the left, RPN in the right)



```
      « → r 'π*r^2' »
                3+5·i
              [3 2 1]
                   r
                 ───
                 m+r
              300000.
 p1 | z1 | R | Q | A12 | α
```



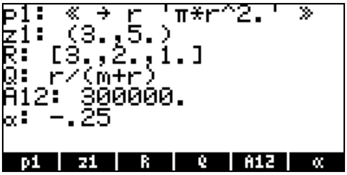
```
6:
5:    « → r 'π*r^2' »
4:              3+5·i
3:            [3 2 1]
2:                 r
                 ───
                 m+r
1:            300000.
 p1 | z1 | R | Q | A12 | α
```

Notice that this time the contents of program *p1* are listed in the screen. To see the remaining variables in this directory, use:

↱ α NXT ↱ A

### Listing the contents of all variables in the screen

Use the keystroke combination ↱ ▼ to list the contents of all variables in the screen. For example:



```
p1: « → r 'π*r^2.' »
z1: (3.,5.)
R: [3.,2.,1.]
Q: r/(m+r)
A12: 300000.
α: -.25
 p1 | z1 | R | Q | A12 | α
```

Press ON to return to normal calculator display.

## Replacing the contents of variables

Replacing the contents of a variable can be thought of as storing a different value in the same variable name. Thus, the examples for creating variables shown above can be used to illustrate the replacement of a variable's content.

### Using the STO▶ command

Using as illustration the six variables, *p1, z1, R, Q, A12, a,* and *A*, created earlier, we will proceed to change the contents of variable *A12* (currently a numerical variable) with the algebraic expression 'β/2', using the STO▶ command. First, using the Algebraic operating mode:

' ALPHA ↱ B ÷ 2 ▶ STO▶ A12 ENTER

Check the new contents of variable *A12* by using ↱ A12 .

Using the RPN operating mode:

' ALPHA ↱ B ÷ 2 ENTER ' A12 ENTER STO▶

or, in a simplified way,

' ALPHA ↱ B ÷ 2 ▶ ' A12 STO▶

## Using the left-shift ↰ key followed by the variable's soft menu key (RPN)

This is a very simple way to change the contents of a variable, but it only works in the RPN mode. The procedure consists in typing the new contents of the variable and entering them into the stack, and then pressing the left-shift key followed by the variable's soft menu key. For example, in RPN, if we want to change the contents of variable *z1* to '*a+b·i* ', use:

' ALPHA ↰ A + ALPHA ↰ B × ↰ i ENTER

This will place the algebraic expression '*a+b·i* ' in level *1:* in the stack. To enter this result into variable *z1*, use: VAR ↰ z1

To check the new contents of *z1*, use: ↱ z1

An equivalent way to do this in Algebraic mode is the following:

ALPHA ↰ A + ALPHA ↰ B × ↰ i ENTER STO▶ z1 ENTER

To check the new contents of *z1*, use: ↱ z1

### Using the ANS(1) variable (Algebraic mode)

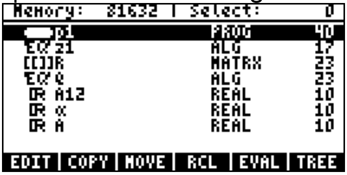
In Algebraic mode one can use the ANS(1) variable to replace the contents of a variable. For example, the procedure for changing the contents of *z1* to '*a+bi*' is the following: ↰ ANS STO▶ z1 ENTER . To check the new contents of *z1*, use: ↱ z1

## Copying variables

The following exercises show different ways of copying variables from one sub-directory to another.

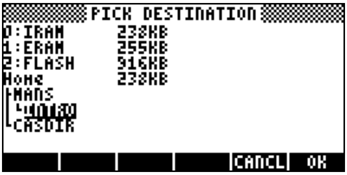
### Using the FILES menu

To copy a variable from one directory to another you can use the FILES menu. For example, within the sub-directory {HOME MANS INTRO}, we have variables *p1, z1, R, Q, A12, α,* and *A*. Suppose that we want to copy variable *A* and place a copy in sub-directory {HOME MANS}. Also, we will copy variable *R* and place a copy in the HOME directory. Here is how to do it: Press ← FILES OK to produce the following list of variables:



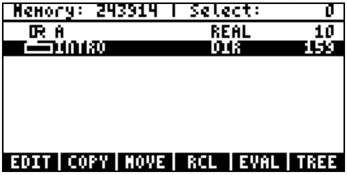
```
Memory:  81632 | Select:      0
   p1                PROG     40
EQ z1                ALG      17
[[]]R                MATRX    23
EQ Q                 ALG      23
 R A12               REAL     10
 R α                 REAL     10
 R A                 REAL     10
EDIT | COPY | MOVE | RCL | EVAL | TREE
```

Use the down-arrow key ▼ to select variable *A* (the last in the list), then press COPY. The calculator will respond with a screen labeled PICK DESTINATION:



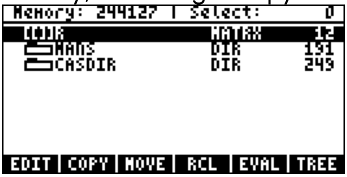
```
PICK DESTINATION
0:IRAM      238KB
1:ERAM      255KB
2:FLASH     916KB
Home        238KB
├MANS
│└INTRO
└CASDIR
              |     |     |     |CANCL| OK
```

Use the up arrow key ▲ to select the sub-directory MANS and press OK. If you now press ← UPDIR , the screen will show the contents of sub-directory MANS (notice that variable A is shown in this list, as expected):



```
Memory: 243914 | Select:      0
 R A                 REAL     10
   INTRO             DIR     159
EDIT | COPY | MOVE | RCL | EVAL | TREE
```

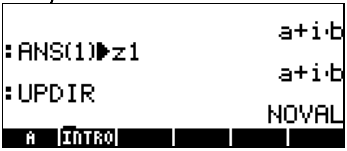
Press ON INTRO ENTER (Algebraic mode), or ON INTRO (RPN mode) to return to the INTRO directory. Press ← FILES OK to produce the list of variables in {HOME MANS INTRO}. Use the down arrow key (▼) to select variable *R*, then press COPY. Use the up arrow key (▲) to select the HOME directory, and press OK. If you now press ← UPDIR , twice, the screen will show the contents of the HOME directory, including a copy of variable *R*:



```
Memory: 244127 | Select:      0
[[]]R                MATRX    12
   MANS              DIR     191
   CASDIR            DIR     249
EDIT | COPY | MOVE | RCL | EVAL | TREE
```

### Using the history in Algebraic mode

Here is a way to use the history (stack) to copy a variable from one directory to another with the calculator set to the Algebraic mode. Suppose that we are within the sub-directory {HOME MANS INTRO}, and want to copy the contents of variable *z1* to sub-directory {HOME MANS}. Use the following procedure: ↱ z1 STO▶ z1 ENTER This simply stores the contents of *z1* into itself (no change effected on *z1*). Next, use ↰ UPDIR ENTER to move to the {HOME MANS} sub-directory. The calculator screen will look like this:

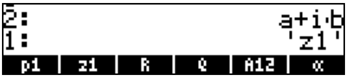


```
                      a+i·b
:ANS(1)▶z1
                      a+i·b
:UPDIR
                      NOVAL
```

Next, use the delete key three times, to remove the last three lines in the display: ← ← ←. At this point, the stack is ready to execute the command ANS(1)▶z1. Press ENTER to execute this command. Then, use ↱ z1, to verify the contents of the variable.

### Using the stack in RPN mode

To demonstrate the use of the stack in RPN mode to copy a variable from one sub-directory to another, we assume you are within sub-directory {HOME MANS INTRO}, and that we will copy the contents of variable *z1* into the HOME directory. Use the following procedure: ↱ z1 ENTER ' z1 ENTER This procedure lists the contents and the name of the variable in the stack. The calculator screen will look like this:



```
2:                    a+i·b
1:                     'z1'
```

Now, use ↰ UPDIR ↰ UPDIR to move to the HOME directory, and press STO▶ to complete the operation. Use ↱ z1, to verify the contents of the variable.

### Copying two or more variables using the stack in Algebraic mode

The following is an exercise to demonstrate how to copy two or more variables using the stack when the calculator is in Algebraic mode. Suppose, once more, that we are within sub-directory {HOME MANS INTRO} and that we want to copy the variables *R* and *Q* into sub-directory {HOME MANS}. The keystrokes necessary to complete this operation are shown following:

↱ R STO▶ R ENTER

↱ Q STO▶ Q ENTER
↰ UPDIR ENTER
◀ ◀ ◀ ENTER
◀ ◀ ◀ ◀ ENTER

To verify the contents of the variables, use ↱ R and ↱ Q.
This procedure can be generalized to the copying of three or more variables.

### Copying two or more variables using the stack in RPN mode

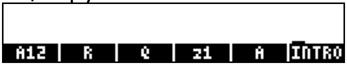
The following is an exercise to demonstrate how to copy two or more variables using the stack when the calculator is in RPN mode. We assume, again, that we are within sub-directory {HOME MANS INTRO} and that we want to copy the variables *R* and *Q* into sub-directory {HOME MANS}. The keystrokes necessary to complete this operation are shown following:

↱ R ' R ENTER
↱ Q ' Q ENTER
↰ UPDIR STO▶ STO▶

To verify the contents of the variables, use ↱ R and ↱ Q.
This procedure can be generalized to the copying of three or more variables.

## Reordering variables in a directory

In this section we illustrate the use of the ORDER command to reorder the variables in a directory. We assume we start within the sub-directory {HOME MANS} containing the variables, *A12, R, Q, z1, A,* and the sub-directory *INTRO*, as shown below. (Copy A12 from INTRO into MANS).

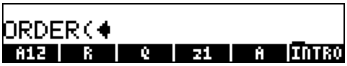


### Algebraic mode

In this case, we have the calculator set to Algebraic mode. Suppose that we want to change the order of the variables to *INTRO, A, z1, Q, R, A12*. Proceed as follows to activate the ORDER function:

| | |
|---|---|
| ↰ PRG ▼ OK | Select MEMORY from the programming menu |
| ▼ ▼ ▼ ▼ OK | Select DIRECTORY from the MEMORY menu |
| ▲ ▲ OK | Select ORDER from the DIRECTORY menu |

The screen will show the following input line:

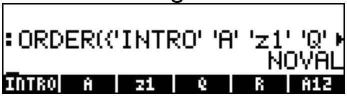


ORDER(◆

Next, we'll list the new order of the variables by using their names typed between quotes:

← {} ' INTRO ▶ → , ' A
▶ → , ' z1 ▶ → , ' Q ▶
→ , ' R ▶ → , ' A12 ENTER

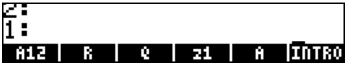
The screen now shows the new ordering of the variables:



:ORDER({'INTRO' 'A' 'z1' 'Q' ▸
NOVAL

### RPN mode

In RPN mode, the list of re-ordered variables is listed in the stack before applying the command ORDER. Suppose that we start from the same situation as above, but in RPN mode, i.e.,



2:
1:

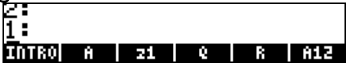
The reordered list is created by using:

← {} INTRO A z1 Q R A12 ENTER

Then, enter the command ORDER, as done before, i.e.,

← PRG ▼ OK Select MEMORY from the programming menu
▼ ▼ ▼ ▼ OK Select DIRECTORY from the MEMORY menu
▲ ▲ OK Select ORDER from the DIRECTORY menu

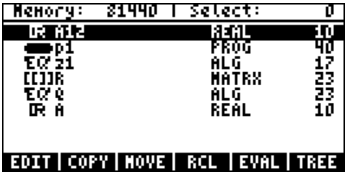
The result is the following screen:



2:
1:

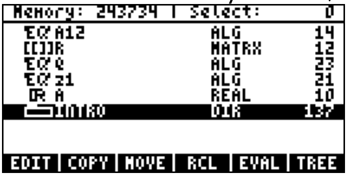
## Moving variables using the FILES menu

To move a variable from one directory to another you can use the FILES menu. For example, within the sub-directory {HOME MANS INTRO}, we have variables *p1, z1, R, Q, A12, α,* and *A*. Suppose that we want to move variable *A12* to sub-directory {HOME MANS}. Here is how to do it: Press ← FILES OK to show a variable list. Use the down-arrow key ▼ to select variable *A12*, then press MOVE. The calculator will respond with a PICK DESTINATION screen. Use the up arrow key ▲ to select the sub-directory MANS and press OK. The screen will now show the contents of sub-directory {HOME MANS INTRO}:



```
Memory:  81440 | Select:    0
   A12              REAL     10
   p1               PROG     40
   z1               ALG      17
 [[]]R              MATRX    23
   Q                ALG      23
   A                REAL     10
EDIT | COPY | MOVE | RCL | EVAL | TREE
```

Notice that variable A12 is no longer there. If you now press ← UPDIR , the screen will show the contents of sub-directory MANS, including variable *A12:*



```
Memory: 243734 | Select:    0
   A12              ALG      14
 [[]]R              MATRX    12
   Q                ALG      23
   z1               ALG      21
   A                REAL     10
   INTRO            DIR     137
EDIT | COPY | MOVE | RCL | EVAL | TREE
```

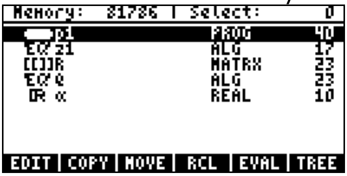
**Note:** You can use the stack to move a variable by combining copying with deleting a variable. Procedures for deleting variables are demonstrated in the next section.

## Deleting variables

Variables can be deleted using function PURGE. This function can be accessed directly by using the TOOLS menu (TOOL), or by using the FILES menu ← FILES OK .

### Using the FILES command

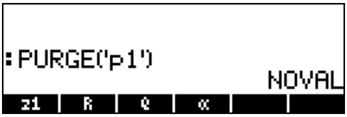
The FILES command can be used to purge one variable at a time. To delete a variable from a given directory you can use the FILES menu. For example, within the sub-directory {HOME MANS INTRO}, we have variables *p1, z1, R, Q, $\alpha$,* and *A* left. Suppose that we delete variable *A*. Here is how to do it: Press ← FILES OK to produce the variable list. Use the down-arrow key ▼ to select variable *A* (the last in the list), then press NXT PURGE YES. The screen will now show the contents of sub-directory INTRO without variable A.



```
Memory:  81786 | Select:    0
   p1               PROG     40
   z1               ALG      17
 [[]]R              MATRX    23
   Q                ALG      23
   α                REAL     10
EDIT | COPY | MOVE | RCL | EVAL | TREE
```

## Using function PURGE in the stack in Algebraic mode

We start again at subdirectory {HOME MANS INTRO} containing now only variables *p1, z1, Q, R,* and $\alpha$. We will use command PURGE to delete variable *p1*. Press TOOL PURGE VAR p1 ENTER. The screen will now show variable *p1* removed:

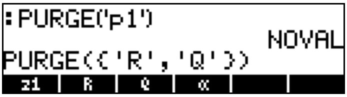


```
:PURGE('p1')
                     NOVAL
 z1 | R | Q | α |   |
```

You can use the PURGE command to erase more than one variable by placing their names in a list in the argument of PURGE. For example, if now we wanted to purge variables *R* and *Q*, simultaneously, we can try the following exercise. Press :

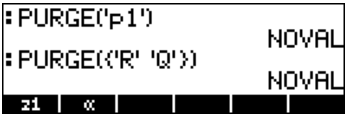
TOOL PURGE ← {} ' VAR R ▶ → , ' VAR Q

At this point, the screen will show the following command ready to be executed:



```
:PURGE('p1')
                     NOVAL
PURGE({'R','Q'})
 z1 | R | Q | α |   |
```

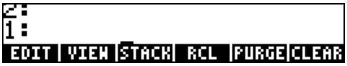
To finish deleting the variables, press ENTER. The screen will now show the remaining variables:



```
:PURGE('p1')
                     NOVAL
:PURGE({'R' 'Q'})
                     NOVAL
 z1 | α |   |   |   |
```

### Using function PURGE in the stack in RPN mode

We start again at subdirectory {HOME MANS INTRO} containing variables *p1, z1, Q, R,* and $\alpha$. We will use command PURGE to delete variable *p1*. Press ' p1 ENTER TOOL PURGE. The screen will now show variable *p1* removed:



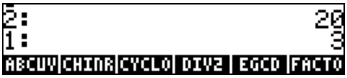
```
2:
1:
EDIT |VIEW |STACK| RCL |PURGE|CLEAR
```

To delete two variables simultaneously, say variables *R* and *Q*, first create a list (in RPN mode, the elements of the list need not be separated by commas as in Algebraic mode): VAR ← {} ' R ▶ ' Q ENTER. Then, press TOOL PURGE use to purge the variables.

# UNDO and CMD functions

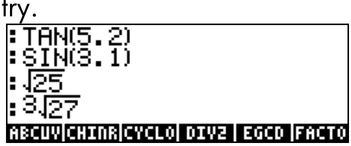
Functions UNDO and CMD are useful for recovering recent commands, or to revert an operation if a mistake was made. These functions are associated with the HIST key: UNDO results from the keystroke sequence ↱ UNDO , while CMD results from the keystroke sequence ↰ CMD .

To illustrate the use of UNDO, try the following exercise in algebraic (ALG) mode: 5 × 4 ÷ 3 ENTER . The UNDO command (↱ UNDO ) will simply erase the result. The same exercise in RPN mode, will follow these keystrokes: 5 ENTER 4 ENTER × 3 ENTER ÷ . Using ↱ UNDO at this point will undo the most recent operation (20/3), leaving the original terms back in the stack:



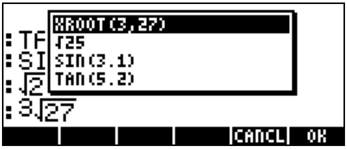
```
2:                        20
1:                         3
ABCUV CHINR CYCLO DIV2 EGCD FACTO
```

To illustrate the use of CMD, let's enter the following entries in ALG mode. Press ENTER after each entry.



```
:TAN(5.2)
:SIN(3.1)
:√25
:³√27
ABCUV CHINR CYCLO DIV2 EGCD FACTO
```

Next, use the CMD function (↰ CMD ) to show the four most recent commands entered by the user, i.e.,



```
XROOT(3,27)
√25
SIN(3.1)
TAN(5.2)
                         CANCL  OK
```

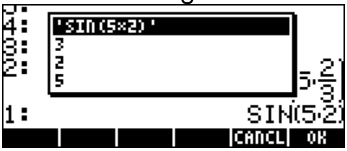
You can use the up and down arrow keys (▲ ▼) to navigate through these commands and highlight any of them that you want to entry anew. Once you have selected the command to enter, press OK.

The CMD function operates in the same fashion when the calculator is in RPN mode, except that the list of commands only shows numbers or algebraics. It does not show functions entered. For example, try the following exercise in RPN mode:

5 ENTER 2 ENTER 3 ÷ × SIN

' SIN 5 × 2 ENTER .

Pressing ← CMD produces the following selection box:



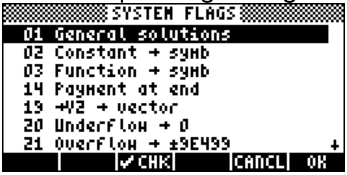
```
4:  'SIN(5×2)'
3:  3
2:  2
    5
1:                SIN(5·2)
                 CANCL  OK
```

As  you can see, the numbers 3, 2, and 5, used in the first calculation above, are listed in the selection box, as well as the algebraic 'SIN(5x2)', but not the SIN function entered previous to the algebraic.

## Flags

A flag is a Boolean value, that can be set or cleared (true or false), that specifies a given setting of the calculator or an option in a program.  Flags in the calculator are identified by numbers.  There are 256 flags, numbered from -128 to 128.  Positive flags are called user flags and are available for programming purposes by the user.  Flags represented by negative numbers are called system flags and affect the way the calculator operates.

To see the current system flag setting press the MODE button, and then the FLAGS soft menu key (i.e., F1).  You will get a screen labeled *SYSTEM FLAGS* listing flag numbers and the corresponding setting.



```
SYSTEM FLAGS
01 General solutions
02 Constant → symb
03 Function → symb
14 Payment at end
19 →V2 → vector
20 Underflow → 0
21 Overflow → ±9E499
        ✓CHK    CANCL  OK
```

(**Note**: In this screen, as only system flags are present, only the absolute value of the flag number Is displayed).  A flag is said to be *set* if you see a check mark (✓) in front of the flag number.  Otherwise, the flag is *not set* or *cleared*. To change the status of a system flag press the CHECK! soft menu key while the flag you want to change is highlighted, or use the +/- key.  You can use the up and down arrow keys (▲▼) to move about the list of system flags.

Although there are 128 system flags, not all of them are used, and some of them are used for internal system control.  System flags that are not accessible to the user are not visible in this screen.  A complete list of flags is presented in Chapter 24.

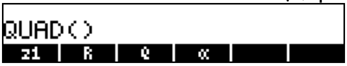
## Example of flag setting: general solutions vs. principal value

For example, the default value for system flag 01 is *General solutions*. What this means is that, if an equation has multiple solutions, all the solutions will be returned by the calculator, most likely in a list. By pressing the CHECK soft menu key you can change system flag 01 to *Principal value*. This setting will force the calculator to provide a single value known as the principal value of the solution.

To see this at work, first set system flag 01 (i.e., select *Principal Value*). Press OK twice to return to normal calculator display. We will try solving a quadratic equation solution, say, $t^2+5t+6 = 0$, with command QUAD.

### *Algebraic mode*

Use the following keystroke sequence: ↱ CAT ALPHA Q (use the up and down arrow keys, ▲ ▼ , to select command QUAD) , press OK .

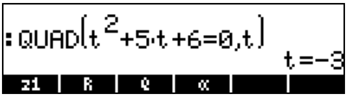


```
QUAD()
```

To enter the equation as the first argument of function QUAD, use the following keystrokes:

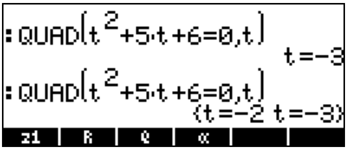
↱ EQW ALPHA ↰ T yˣ 2 ▶ + 5 × ALPHA ↰ T + 6 ▲ ▲
↱ = 0 ENTER
↱ , ALPHA ↰ T ENTER

The result is:



```
:QUAD(t^2+5·t+6=0,t)
                          t=-3
```

Now, change the setting of flag 1 to *General solutions*: MODE FLAGS CHECK OK OK . And try the solution again: ▲ ▲ ENTER ENTER . The solution now includes two values:



```
:QUAD(t^2+5·t+6=0,t)
                          t=-3
:QUAD(t^2+5·t+6=0,t)
                 {t=-2 t=-3}
```

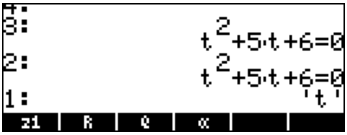
### *RPN mode*

First set system flag 01 (i.e., *Principal Value*). Press OK twice to return to normal calculator display. Then, type the quadratic equation as follows:

↱ EQW ALPHA ↰ T Y^x 2 ▶ + 5 × ALPHA ↰ T + 6 ▲ ▲
↱ = 0 ENTER

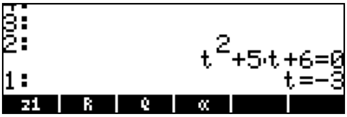
ENTER (keeping a second copy in the RPN stack)

' ALPHA ↰ T ENTER



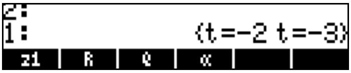
```
4:
3:                  t^2+5·t+6=0
2:                  t^2+5·t+6=0
1:                          't'
```

Use the following keystroke sequence to enter the QUAD command: ↱ CAT ALPHA Q (use the up and down arrow keys, ▲ ▼ , to select command QUAD) , press OK . The screen shows the principal solution:



```
3:
2:                  t^2+5·t+6=0
1:                         t=-3
```

Now, change the setting of flag 01 to *General solutions*: MODE FLAGS CHECK OK OK. And try the solution again: ◀ ' ALPHA ↰ T ENTER ↱ CAT ALPHA Q (use the up and down arrow keys, ▲ ▼ , to select command QUAD) , press OK . The screen now shows the two solutions:



```
2:
1:               {t=-2 t=-3}
```

## Other flags of interest

Bring up once more the current flag setting by pressing the MODE button, and then the FLAGS soft menu key. Make sure to clear system flag 01, which was left set from the previous exercise. Use the up and down arrow keys (▲ ▼) to move about the system flag list.

Some flags of interest and their preferred value for the purpose of the exercises that follow in this manual are:

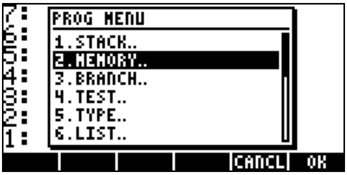
| | |
|---|---|
| *02 Constant → symb*: | Constant values (e.g., π) are kept as symbols |
| *03 Function → symb*: | Functions are not automatically evaluated, instead they are loaded as symbolic expressions. |
| *27 'X+Y*i' → (X,Y):* | Complex numbers are represented as ordered pairs |
| *60 [α][α] locks*: | The sequence ALPHA ALPHA locks the alphabetic keyboard |

Press OK twice to return to normal calculator display.

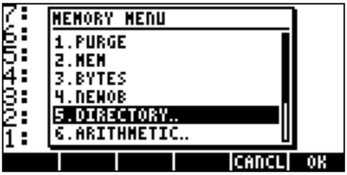
## CHOOSE boxes vs. Soft MENU

In some of the exercises presented in this chapter we have seen menu lists of commands displayed in the screen. This menu lists are referred to as *CHOOSE boxes*. For example, to use the ORDER command to reorder variables in a directory, we used:

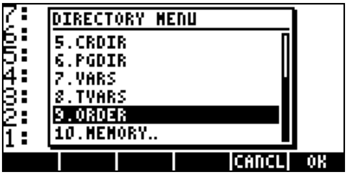
⬅ PRG ▼ Show PROG menu list and select MEMORY



OK ▼▼▼▼ Show the MEMORY menu list and select DIRECTORY



OK ▲▲ Show the DIRECTORY menu list and select ORDER

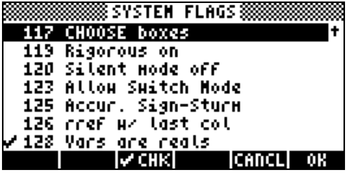


OK activate the ORDER command

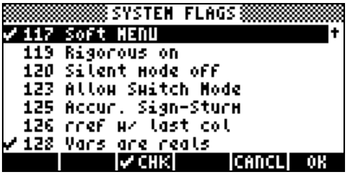
There is an alternative way to access these menus as *soft MENU* keys, by setting flag *117*. To set this flag try the following:

MODE FLAGS ▲▲▲▲▲▲▲

The screen shows flag 117 not set (*CHOOSE boxes*), as shown here:


SYSTEM FLAGS
117 CHOOSE boxes
119 Rigorous on
120 Silent mode off
123 Allow Switch Mode
125 Accur. Sign-Sturm
126 rref w/ last col
✓128 Vars are reals

Press the CHECK soft menu key to set flag 117 to *soft MENU*. The screen will reflect that change:


SYSTEM FLAGS

✓ 117 Soft MENU

119 Rigorous on

120 Silent mode off

123 Allow Switch Mode

125 Accur. Sign-Sturm

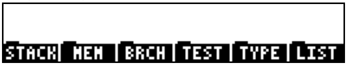
126 rref w/ last col

✓ 128 Vars are reals

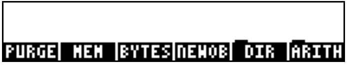
Press twice to return to normal calculator display.

Now, we'll try to find the ORDER command using similar keystrokes to those used above, i.e., we start with ← PRG .

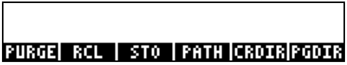
Notice that instead of a menu list, we get soft menu labels with the different options in the PROG menu, i.e.,



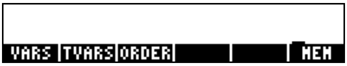
Press F2 to select the MEMORY soft menu (MEM). The display now shows:



Press F5 to select the DIRECTORY soft menu (DIR)



The ORDER command is not shown in this screen. To find it we use the NXT key to find it:

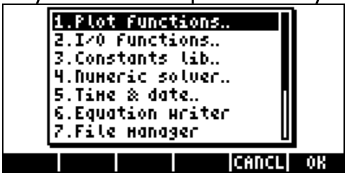


To activate the ORDER command we press the F3 (ORDER) soft menu key. Although not applied to a specific example, this exercise shows the two options for menus in the calculator (CHOOSE boxes and soft MENUs).

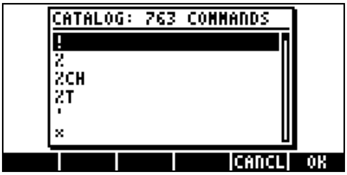
## Selected CHOOSE boxes

Some menus will only produce CHOOSE boxes, e.g.,

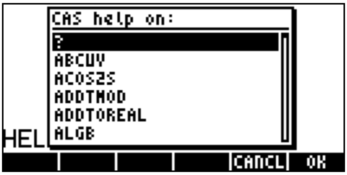
- The APPS (APPlicationS menu), activated with the APPS key, first key in the second row of keys from the top of the keyboard:



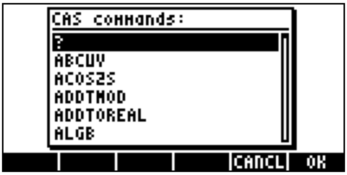
- The CAT (CATalog menu), activated with the ↱ CAT key, second key in the fourth row of keys from the top of the keyboard:



- The HELP menu, activated with TOOL NXT HELP



- The CMDS (CoMmanDS) menu, activated within the Equation Writer, i.e., ⇨ EQW NXT CMDS



# Chapter 3
# Calculation with real numbers

This chapter demonstrates the use of the calculator for operations and functions related to real numbers.  Operations along these lines are useful for most common calculations in the physical sciences and engineering.  The user should be acquainted with the keyboard to identify certain functions available in the keyboard (e.g., SIN, COS, TAN, etc.).  Also, it is assumed that the reader knows how to adjust the calculator's operation, i.e., select operating mode (see Chapter 1), use menus and choose boxes (see Chapter 1), and operate with variables (see Chapter 2).

## Checking calculators settings

To check the current calculator and CAS settings you need to just look at the top line in the calculator display in normal operation.  For example, you may see the following setting: RAD XYZ DEC **R** = 'X'

This stands for RADians for angular measurements, XYZ for Rectangular (Cartesian) coordinates, DECimal number base, **R**eal numbers preferred, = means "exact" results, and 'X' is the value of the default independent variable.

Another possible listing of options could be DEG R∠Z HEX C ~ 't'

This stands for DEGrees as angular measurements, R∠Z for Polar coordinates, HEXagesimal number base, Complex numbers allowed, ~ stands for "approximate" results, and 't' as the default independent variable.

In general, this part of the display contains seven elements. Each element is identified next under numbers 1 through 7.  The possible values for each element are shown between parentheses following the element description. The explanation of each of those values is also shown:

1. Angle measure specification (DEG, RAD, GRD)
   DEG: degrees, 360 degrees in a complete circle
   RAD: radians, 2π radians in a complete circle
   GRD: grades, 400 grades in a complete circle

2. Coordinate system specification (XYZ, R∠Z, R∠∠). The symbol ∠ stands for an angular coordinate.
   XYZ: Cartesian or rectangular (x,y,z)
   R∠Z: cylindrical Polar coordinates (r,θ,z)
   R∠∠: Spherical coordinates (ρ,θ,φ)
3. Number base specification (HEX, DEC, OCT, BIN)
   HEX: hexadecimal numbers (base 16)
   DEC: decimal numbers (base 10)
   OCT: octal numbers (base 8)
   BIN: binary numbers (base 2)
4. Real or complex mode specification (**R**, **C**)
   **R**: real numbers
   **C**: complex numbers
5. Exact or approximate mode specification (=, ~)
   = exact (symbolic) mode
   ~ approximate (numerical) mode
6. Default CAS independent variable (e.g., 'X', 't', etc.)

### Checking calculator mode

When in RPN mode the different levels of the stack are listed in the left-hand side of the screen. When the ALGEBRAIC mode is selected there are no numbered stack levels, and the word ALG is listed in the top line of the display towards the right-hand side. The difference between these operating modes was described in detail in Chapter 1.

## Real number calculations

To perform real number calculations it is preferred to have the CAS set to *Real* (as opposite to *Complex*) mode. In some cases, a complex result may show up, and a request to change the mode to *Complex* will be made by the calculator. *Exact* mode is the default mode for most operations. Therefore, you may want to start your calculations in this mode. Any change to *Approx* mode required to complete an operation will be requested by the calculator. There is no preferred selection for the angle measure or for the number base specification. Real number calculations will be demonstrated in both the Algebraic (ALG) and Reverse Polish Notation (RPN) modes.

## Changing sign of a number, variable, or expression

Use the +/- key. In ALG mode, you can press +/- before entering the number, e.g., +/- 2 • 5 ENTER. Result = -2.5. In RPN mode, you need to enter at least part of the number first, and then use the +/- key, e.g., 2 • 5 +/-. Result = -2.5. If you use the +/- function while there is no command line, the calculator will apply the NEG function (inverse of sign) to the object on the first level of the stack.

## The inverse function

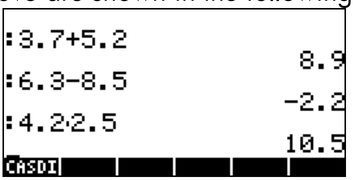
Use the 1/x key. In ALG mode, press 1/x first, followed by a number or algebraic expression, e.g., 1/x 2. Result = ½ or 0.5. In RPN mode, enter the number first, then use the key, e.g., 4 ENTER 1/x. Result = ¼ or 0.25.

## Addition, subtraction, multiplication, division

Use the proper operation key, namely, + − × ÷. In ALG mode, press an operand, then an operator, then an operand, followed by an ENTER to obtain a result. Examples:

3 • 7 + 5 • 2 ENTER
6 • 3 − 8 • 5 ENTER
4 • 2 × 2 • 5 ENTER
2 • 3 ÷ 4 • 5 ENTER

The first three operations above are shown in the following screen shot:



```
:3.7+5.2
                 8.9
:6.3-8.5
                -2.2
:4.2·2.5
                10.5
CASDI
```

In RPN mode, enter the operands one after the other, separated by an ENTER, then press the operator key. Examples:

3 • 7 ENTER 5 • 2 +
6 • 3 ENTER 8 • 5 −
4 • 2 ENTER 2 • 5 ×
2 • 3 ENTER 4 • 5 ÷

Alternatively, in RPN mode, you can separate the operands with a space (SPC) before pressing the operator key. Examples:

3 • 7 SPC 5 • 2 +

6 • 3 SPC 8 • 5 −
4 • 2 SPC 2 • 5 ×
2 • 3 SPC 4 • 5 ÷

## Using parentheses

Parentheses can be used to group operations, as well as to enclose arguments of functions. The parentheses are available through the keystroke combination ↰ ( ) . Parentheses are always entered in pairs. For example, to calculate (5+3.2)/(7-2.2):

In ALG mode:
↰ ( ) 5 + 3 • 2 ▶ ÷ ↰ ( ) 7 − 2 • 2 ENTER

In RPN mode, you do not need the parenthesis, calculation is done directly on the stack:
5 ENTER 3 • 2 ENTER + 7 ENTER 2 • 2 ENTER − ÷

In RPN mode, typing the expression between quotes will allow you to enter the expression like in algebraic mode:
' ↰ ( ) 5 + 3 • 2 ▶ ÷
↰ ( ) 7 − 2 • 2 ENTER EVAL

For both, ALG and RPN modes, using the Equation Writer:
↱ EQW 5 + 3 • 2 ▶ ÷ 7 − 2 • 2

The expression can be evaluated within the Equation writer, by using
▲ ▲ ▲ ▲ EVAL or, ↱ ▲ EVAL

## Absolute value function

The absolute value function, ABS, is available through the keystroke combination: ↰ ABS . When calculating in the stack in ALG mode, enter the function before the argument, e.g., ↰ ABS +/- 2 • 3 2 ENTER

In RPN mode, enter the number first, then the function, e.g.,
2 • 3 2 +/- ↰ ABS

## Squares and square roots

The square function, SQ, is available through the keystroke combination: ↰ $x^2$ . When calculating in the stack in ALG mode, enter the function before the argument, e.g., ↰ $x^2$ +/- 2 • 3 ENTER

In RPN mode, enter the number first, then the function, e.g.,
2 • 3 +/- ↰ $x^2$

The square root function, √, is available through the R key. When calculating in the stack in ALG mode, enter the function before the argument, e.g.,
√x 1 2 3 • 4 ENTER

In RPN mode, enter the number first, then the function, e.g.,
1 2 3 • 4 √x

## Powers and roots

The power function, ^, is available through the $Y^x$ key. When calculating in the stack in ALG mode, enter the base (*y*) followed by the $Y^x$ key, and then the exponent (*x*), e.g., 5 • 2 $Y^x$ 1 • 2 5

In RPN mode, enter the number first, then the function, e.g.,
5 • 2 ENTER 1 • 2 5 ENTER $Y^x$

The root function, XROOT*(y,x),* is available through the keystroke combination ↱ $\sqrt[x]{y}$ . When calculating in the stack in ALG mode, enter the function XROOT followed by the arguments (*y*,*x*), separated by commas, e.g.,
↱ $\sqrt[x]{y}$ 3 ↱ , 2 7 ENTER

In RPN mode, enter the argument *y*, first, then, *x*, and finally the function call, e.g., 2 7 ENTER 3 ENTER ↱ $\sqrt[x]{y}$

## Base-10 logarithms and powers of 10

Logarithms of base 10 are calculated by the keystroke combination ↱ LOG (function LOG) while its inverse function (ALOG, or antilogarithm) is calculated by using ↰ $10^x$ . In ALG mode, the function is entered before the argument:
↱ LOG 2 • 4 5 ENTER
↰ $10^x$ +/- 2 • 3 ENTER

In RPN mode, the argument is entered before the function
2 • 4 5 ENTER ↱ LOG
2 • 3 +/- ENTER ↰ $10^x$

## Using powers of 10 in entering data

Powers of ten, i.e., numbers of the form $-4.5\times10^{-2}$, etc., are entered by using the EEX key. For example, in ALG mode:

+/- 4 • 5 EEX +/- 2 ENTER

Or, in RPN mode:

4 • 5 +/- EEX 2 +/- ENTER

## Natural logarithms and exponential function

Natural logarithms (i.e., logarithms of base *e = 2.7182818282*) are calculated by the keystroke combination ↱ LN (function LN) while its inverse function, the exponential function (function EXP) is calculated by using ↰ $e^x$ . In ALG mode, the function is entered before the argument:

↱ LN 2 • 4 5 ENTER

↰ $e^x$ +/- 2 • 3 ENTER

In RPN mode, the argument is entered before the function

2 • 4 5 ENTER ↱ LN

2 • 3 +/- ENTER ↰ $e^x$

## Trigonometric functions

Three trigonometric functions are readily available in the keyboard: sine (SIN), cosine (COS), and tangent (TAN). The arguments of these functions are angles, therefore, they can be entered in any system of angular measure (degrees, radians, grades). For example, with the DEG option selected, we can calculate the following trigonometric functions:

In ALG mode:

SIN 3 0 ENTER

COS 4 5 ENTER

TAN 1 3 5 ENTER

In RPN mode:

3 0 ENTER SIN

4 5 ENTER COS

1 3 5 ENTER TAN

## Inverse trigonometric functions

The inverse trigonometric functions available in the keyboard are the arcsine (ASIN), arccosine (ACOS), and arctangent (ATAN), available through the keystroke combinations ↰ ASIN , ↰ ACOS , and ↰ ATAN , respectively. Since the inverse trigonometric functions represent angles, the answer from these functions will be given in the selected angular measure (DEG, RAD, GRD). Some examples are shown next:

In ALG mode:

↰ ASIN 0 • 2 5 ENTER
↰ ACOS 0 • 8 5 ENTER
↰ ATAN 1 • 3 5 ENTER

In RPN mode:

0 • 2 5 ENTER ↰ ASIN
0 • 8 5 ENTER ↰ ACOS
1 • 3 5 ENTER ↰ ATAN

All the functions described above, namely, ABS, ABS, SQ, √, ^, XROOT, LOG, ALOG, LN, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN, can be combined with the fundamental operations (+ − × ÷) to form more complex expressions. The Equation Writer, whose operations is described in Chapter 2, is ideal for building such expressions, regardless of the calculator operation mode.

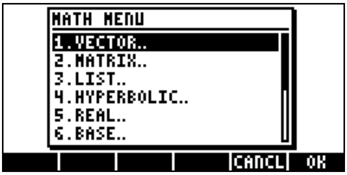
## Differences between functions and operators

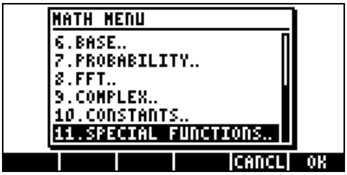
Functions like ABS, ABS, SQ, √, LOG, ALOG, LN, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN require a single argument. Thus, their application is ALG mode is straightforward, e.g., ABS(x). Some functions like XROOT require two arguments, e.g., XROOT(x,y). This function has the equivalent keystroke sequence ↱ $\sqrt[x]{y}$ .

Operators, on the other hand, are placed after a single argument or between two arguments. The factorial operator (!), for example, is placed after a number, e.g., 5 ALPHA ↱ 2 ENTER . Since this operator requires a single argument it is referred to as a unary operator. Operators that require two arguments, such as + − × ÷ $Y^x$, are binary operators, e.g., 3 × 5 , or 4 $Y^x$ 2 .

# Real number functions in the MTH menu

The MTH (MaTHematics) menu include a number of mathematical functions mostly applicable to real numbers. To access the MTH menu, use the keystroke combination ↰ MTH . With the default setting of *CHOOSE boxes* for system flag 117 (see Chapter 2), the MTH menu is shown as the following menu list:





As they are a great number of mathematic functions available in the calculator, the MTH menu is sorted by the type of object the functions apply on. For example, options 1*. VECTOR.., 2. MATRIX.*, and *3. LIST..* apply to those data types (i.e., vectors, matrices, and lists) and will discussed in more detail in subsequent chapters. Options *4. HYPERBOLIC..* and *5. REAL..* apply to real numbers and will be discussed in detailed herein. Option *6. BASE..* is used for conversion of numbers in different bases, and is also to be discussed in a separate chapter. Option *7. PROBABILITY..* is used for probability applications and will be discussed in an upcoming chapter. Option *8. FFT..* (Fast Fourier Transform) is an application of signal processing and will be discussed in a different chapter. Option *9. COMPLEX..* contains functions appropriate for complex numbers, which will be discussed in the next chapter. Option *10. CONSTANTS* provides access to the constants in the calculator. This option will be presented later in this section. Finally, option *11. SPECIAL FUNCTIONS..* includes functions of advanced mathematics that will be discussed in this section also.

In general, to apply any of these functions you need to be aware of the number and order of the arguments required, and keep in mind that, in ALG mode you should select first the function and then enter the argument, while in RPN mode, you should enter the argument in the stack first, and then select the function.

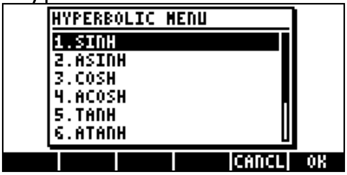
**Using calculator menus**:

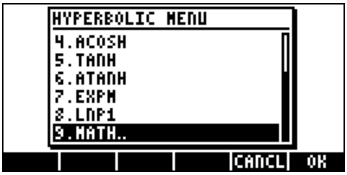
**1.** Since the operation of *MTH* functions (and of many other calculator menus) is very similar, we will describe in detail the use of the *4. HYPERBOLIC..* menu in this section, with the intention of describing the general operation of calculator menus. Pay close attention to the process for selecting different options.

**2.** To quickly select one of the numbered options in a menu list (or CHOOSE box), simply press the number for the option in the keyboard. For example, to select option *4. HYPERBOLIC..* in the MTH menu, simply press 4.

## Hyperbolic functions and their inverses

Selecting Option *4. HYPERBOLIC..* , in the *MTH* menu, and pressing OK, produces the hyperbolic function menu:





The hyperbolic functions are:

Hyperbolic sine, SINH, and its inverse, ASINH or sinh$^{-1}$
Hyperbolic cosine, COSH, and its inverse, ACOSH or cosh$^{-1}$
Hyperbolic tangent, TANH, and its inverse, ATANH or tanh$^{-1}$

This menu contains also the functions:

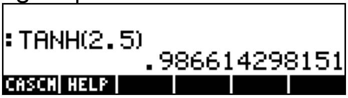
EXPM(x) = exp(x) – 1,
LNP1(x) = ln(x+1).

Finally, option *9. MATH*, returns the user to the MTH menu.

For example, in ALG mode, the keystroke sequence to calculate, say, *tanh(2.5),* is the following:

| | |
|---|---|
| ← *MTH* | Select *MTH* menu |
| 4 OK | Select the *4. HYPERBOLIC..* menu |
| 5 OK | Select the *5. TANH* function |
| 2 • 5 ENTER | Evaluate *tanh(2.5)* |

The screen shows the following output:



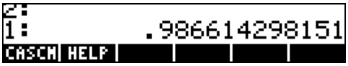
```
:TANH(2.5)
                .986614298151
```

In the RPN mode, the keystrokes to perform this calculation are the following:

| | |
|---|---|
| 2 • 5 ENTER | Enter the argument in the stack |
| ← *MTH* | Select *MTH* menu |

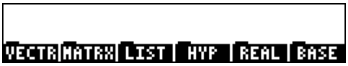
4 OK Select the *4. HYPERBOLIC..* menu
5 OK Select the *5. TANH* function

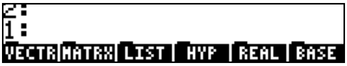
The result is:



```
2:
1:          .986614298151
```

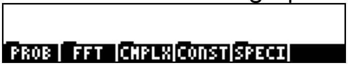
The operations shown above assume that you are using the default setting for system flag 117 (*CHOOSE boxes*). If you have changed the setting of this flag (see Chapter 2) to *SOFT menu*, the MTH menu will show as labels of the soft menu keys, as follows (left-hand side in ALG mode, right –hand side in RPN mode):

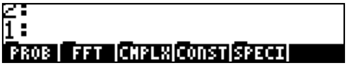




```
2:
1:
```

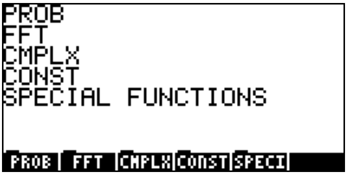
Pressing NXT shows the remaining options:



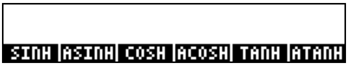


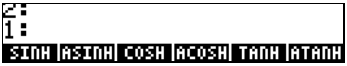
```
2:
1:
```

**Note:** Pressing ← PREV will return to the first set of *MTH* options. Also, using the combination → ▼ will list all menu functions in the screen, e.g.



Thus, to select, for example, the hyperbolic functions menu, with this menu format press HYP , to produce:





```
2:
1:
```

Finally, in order to select, for example, the hyperbolic tangent (tanh) function, simply press TANH.

**Note:** To see additional options in these soft menus, press the NXT key or the ← PREV keystroke sequence.

For example, to calculate *tanh(2.5),* in the ALG mode, when using *SOFT menus* over *CHOOSE boxes*, follow this procedure:

| | |
|---|---|
| ← MTH | Select *MTH* menu |
| HYP | Select the *HYPERBOLIC..* menu |
| TANH | Select the *TANH* function |
| 2 . 5 ENTER | Evaluate *tanh(2.5)* |

In RPN mode, the same value is calculated using:

| | |
|---|---|
| 2 . 5 ENTER | Enter argument in the stack |
| ← MTH | Select *MTH* menu |
| HYP | Select the *HYPERBOLIC..* menu |
| TANH | Select the *TANH* function |

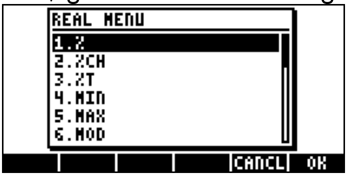
As an exercise of applications of hyperbolic functions, verify the following values:

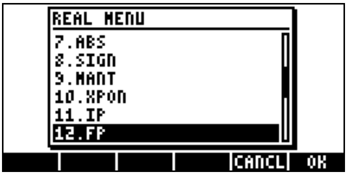
*sinh (2.5) = 6.05020..* *sinh*$^{-1}$*(2.0) = 1.4436…*
*cosh (2.5) = 6.13228..* *acosh*$^{-1}$*(2.0) = 1.3169…*
*tanh (2.5) = 0.98661..* *tanh*$^{-1}$*(0.2) = 0.2027…*
*expm(2.0) = 6.38905….* *lnp1(1) = 0.69314….*

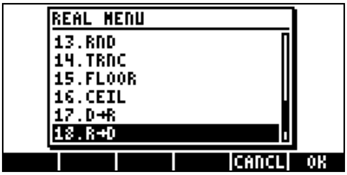
Once again, the general procedure shown in this section can be applied for selecting options in any calculator menu.

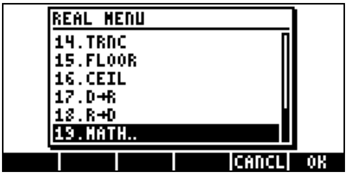
## Real number functions

Selecting option *5. REAL..* in the MTH menu, with system flag 117 set to *CHOOSE boxes*, generates the following menu list:


REAL MENU
1.%
2.%CH
3.%T
4.MIN
5.MAX
6.MOD

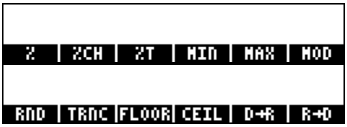

REAL MENU
7.ABS
8.SIGN
9.MANT
10.XPON
11.IP
12.FP

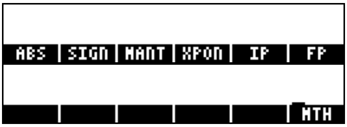




Option *19. MATH..* returns the user to the MTH menu. The remaining functions are grouped into six different groups described below.

If system flag 117 is set to *SOFT menus*, the *REAL* functions menu will look like this (ALG mode used, the same soft menu keys will be available in RPN mode):





The very last option, MTH, returns the user to the *MTH* menu.

**Percentage functions**

These functions are used to calculate percentages and related values as follows:

% (y,x) : calculates the x percentage of y

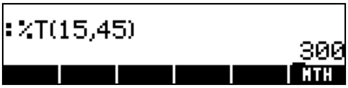
%CH(y,x) : calculates 100(y-x)/x, i.e., the percentage change

%T(y,x) : calculates 100 x/y

These functions require two arguments, we illustrate the calculation of *%T(15,45)*, i.e., calculation 15% of 45, next. We assume that the calculator is set to ALG mode, and that system flag 117 is set to *CHOOSE boxes*. The procedure is as follows:

| | |
|---|---|
| ← MTH | Select *MTH* menu |
| 5 OK | Select the *5. REAL..* menu |
| 3 OK | Select the *5. %T* function |
| 1 5 | Enter first argument |
| → , | Enter a comma to separate arguments |
| 4 5 | Enter second argument |
| ENTER | Calculate function |

The result is shown next:



```
:%T(15,45)
                300
```

In RPN mode, recall that argument *y* is located in the second level of the stack, while argument *x* is located in the first level of the stack. This means, you should enter *x* first, and then, *y*, just as in ALG mode. Thus, the calculation of *%T(15,45)*, in RPN mode, and with system flag 117 set to *CHOOSE boxes*, we proceed as follows:

| | |
|---|---|
| 1 5 ENTER | Enter first argument |
| 4 5 ENTER | Enter second argument |
| ← MTH | Select *MTH* menu |
| 5 OK | Select the *5. REAL..* menu |
| 3 OK | Select the *5. %T* function |

**Note:** The exercises in this section illustrate the general use of calculator functions having 2 arguments. The operation of functions having 3 or more arguments can be generalized from these examples.

As an exercise for percentage-related functions, verify the following values: *%(5,20) = 1, %CH(22,25) = 13.6363.., %T(500,20) = 20*

### Minimum and maximum

Use these functions to determine the minimum or maximum value of two arguments.

MIN(x,y) : minimum value of x and y
MAX(x,y) : maximum value of x and y

As an exercise, verify that *MIN(-2,2) = -2, MAX(-2,2) = 2*

### Modulo

MOD: y mod x = residual of y/x, i.e., if x and y are integer numbers, y/x = d + r/x, where d = quotient, r = residual. In this case, r = y mod x.

Please notice that MOD is not a function, but rather an operator, i.e., in ALG mode, MOD should be used as `y MOD x`, and not as `MOD(y,x)`. Thus, the operation of MOD is similar to that of +, −, ×, ÷.

As an exercise, verify that *15 MOD 4 = 15 mod 4 = residual of 15/4 = 3*

**Absolute value, sign, mantissa, exponent, integer and fractional parts**

ABS(x) : calculates the absolute value, |x|
SIGN(x): determines the sign of x, i.e., -1, 0, or 1.
MANT(x): determines the mantissa of a number based on $\log_{10}$.
XPON(x): determines the power of 10 in the number
IP(x) : determines the integer part of a real number
FP(x) : determines the fractional part of a real number

As an exercise, verify that *ABS(-3) = |-3| = 3, SIGN(-5) = -1, MANT(2540) = 2.540, XPON(2540) = 3, IP(2.35) = 2, FP(2.35) = 0.35.*

**Rounding, truncating, *floor*, and *ceiling* functions**

RND(x,y) : rounds up y to x decimal places
TRNC(x,y) : truncate y to x decimal places
FLOOR(x) : closest integer that is less than or equal to x
CEIL(x) : closest integer that is greater than or equal to x

As an exercise, verify that *RND(1.4567,2) = 1.46, TRNC(1.4567,2) = 1.45, FLOOR(2.3) = 2, CEIL(2,3) = 3*

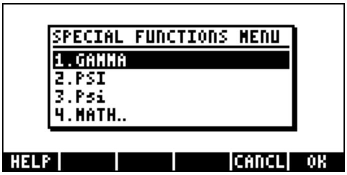
**Radians-to-degrees and degrees-to-radians functions**

D→R (x) : converts degrees to radians
R→D (x) : converts radians to degrees.

As an exercise, verify that *D→R(45) = 0.78539* (i.e., 45° = $0.78539^{rad}$), *R→D(1.5) = 85.943669..* (i.e., $1.5^{rad}$ = 85.943669..°).

## Special functions

Option *11. Special functions…* in the MTH menu includes the following functions:


SPECIAL FUNCTIONS MENU
1.GAMMA
2.PSI
3.Psi
4.MATH..

| | |
|---|---|
| GAMMA: | The Gamma function Γ(α) |
| PSI: | N-th derivative of the digamma function |
| Psi: | Digamma function, derivative of the ln(Gamma) |

The Gamma function is defined by $\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x} dx$. This function has applications in applied mathematics for science and engineering, as well as in probability and statistics.

Factorial of a number

The factorial of a positive integer number *n* is defined as *n!=n·(n-1)·(n-2) …3·2·1*, with *0! = 1*. The factorial function is available in the calculator by using ALPHA ↱ 2. In both ALG and RPN modes, enter the number first, followed by the sequence ALPHA ↱ 2. Example: 5 ALPHA ↱ 2 ENTER.
The Gamma function, defined above, has the property that

$$\Gamma(\alpha) = (\alpha-1)\ \Gamma(\alpha-1), \text{ for } \alpha > 1.$$

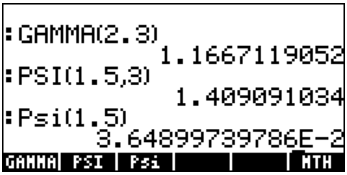
Therefore, it can be related to the factorial of a number, i.e., $\Gamma(\alpha) = (\alpha-1)!$, when α is a positive integer. We can also use the factorial function to calculate the Gamma function, and vice versa. For example, $\Gamma(5) = 4!$ or, 4 ALPHA ↱ 2 ENTER. The factorial function is available in the MTH menu, through the *7. PROBABILITY..* menu.

The PSI function, Ψ*(x,y)*, represents the y-th derivative of the digamma function, i.e., $\Psi(n,x) = \frac{d^n}{dx^n}\psi(x)$, where *ψ(x)* is known as the digamma function, or Psi function. For this function, *y* must be a positive integer.

The Psi function, *ψ(x)*, or digamma function, is defined as $\psi(x) = \ln[\Gamma(x)]$.

Examples of these special functions are shown here using both the ALG and RPN modes. As an exercise, verify that *GAMMA(2.3) = 1.166711…*, *PSI(1.5,3) = 1.40909..*, and *Psi(1.5) = 3.64899739..E-2.*

These calculations are shown in the following screen shot:



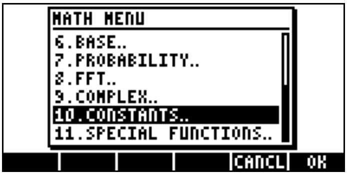
```
:GAMMA(2.3)
              1.1667119052
:PSI(1.5,3)
               1.409091034
:Psi(1.5)
          3.64899739786E-2
```

## Calculator constants

The following are the mathematical constants used by your calculator:

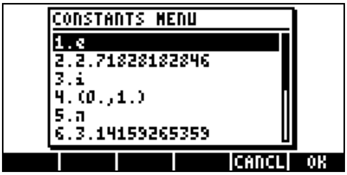
- *e*: the base of natural logarithms.
- *i*: the imaginary unit, $i^2$ = -1.
- *π*: the ratio of the length of the circle to its diameter.
- MINR: the minimum real number available to the calculator.
- MAXR: the maximum real number available to the calculator.

To have access to these constants, select option *11. CONSTANTS..* in the *MTH* menu,

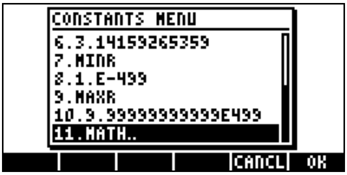


```
MATH MENU
6.BASE..
7.PROBABILITY..
8.FFT..
9.COMPLEX..
10.CONSTANTS..
11.SPECIAL FUNCTIONS..
```

The constants are listed as follows:



```
CONSTANTS MENU
1.e
2.2.71828182846
3.i
4.(0.,1.)
5.π
6.3.14159265359
```



```
CONSTANTS MENU
6.3.14159265359
7.MINR
8.1.E-499
9.MAXR
10.9.99999999999E499
11.MATH..
```

Selecting any of these entries will place the value selected, whether a symbol (e.g., *e*, *i*, *π*, *MINR*, or *MAXR*) or a value (*2.71.., (0,1), 3.14.., 1E-499, 9.99..E499*) in the stack.

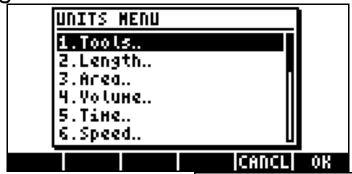
Please notice that *e* is available from the keyboard as *exp(1)*, i.e., ↰ $e^x$ 1 ENTER, in ALG mode, or 1 ENTER ↰ $e^x$, in RPN mode. Also, π is available directly from the keyboard as ↰ π. Finally, *i* is available by using ↰ *i*.

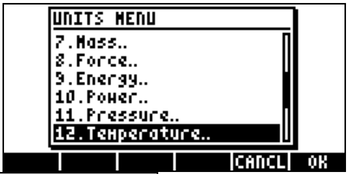
# Operations with units

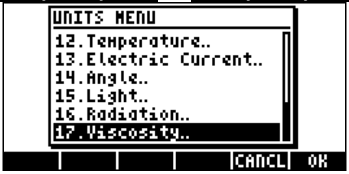
Numbers in the calculator can have units associated with them. Thus, it is possible to calculate results involving a consistent system of units and produce a result with the appropriate combination of units.

## The UNITS menu

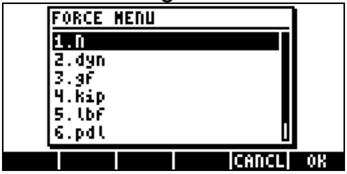
The units menu is launched by the keystroke combination ↱ UNITS (associated with the 6 key). With system flag 117 set to *CHOOSE boxes*, the result is the following menu:

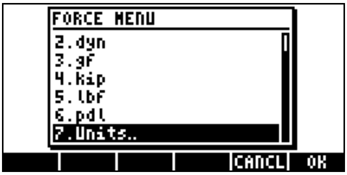






Option *1. Tools..* contains functions used to operate on  units (discussed later). Options *3. Length..*  through *17.Viscosity..* contain menus with a number of units for each of the quantities described. For example, selecting option *8. Force..* shows the following units menu:

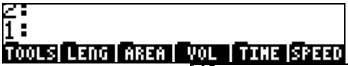


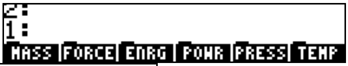


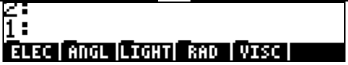
The user will recognize most of these units (some, e.g., dyne, are not used very often nowadays) from his or  her physics classes: *N*  = newtons, *dyn* = dynes, *gf* = grams – force (to distinguish from gram-mass, or plainly gram, a unit of mass), *kip* = kilo-poundal (1000 pounds), *lbf* = pound-force (to distinguish from pound-mass), *pdl* = poundal.

To attach a unit object to a number, the number must be followed by an underscore. Thus, a force of 5 N will be entered as 5_N.

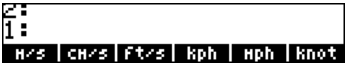
For extensive operations with units SOFT menus provide a more convenient way of attaching units. Change system flag 117 to SOFT menus (see Chapter 1), and use the keystroke combination → UNITS to get the following menus. Press NXT to move to the next menu page.

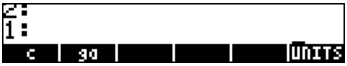






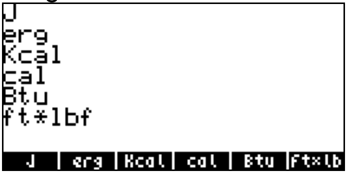
Pressing on the appropriate soft menu key will open the sub-menu of units for that particular selection. For example, for the SPEED sub-menu, the following units are available:

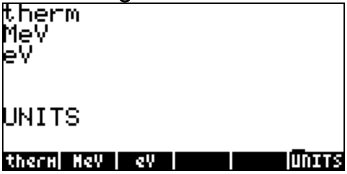




Pressing the soft menu key UNITS will take you back to the UNITS menu.

Recall that you can always list the full menu labels in the screen by using → ▼, e.g., for the ENRG set of units the following labels will be listed:





**Note:** Use the NXT key or the ← PREV keystroke sequence to navigate through the menus.

## Available units

The following is a list of the units available in the UNITS menu. The unit symbol is shown first followed by the unit name in parentheses:

***LENGTH***

m (meter), cm (centimeter), mm (millimeter), yd (yard), ft (feet), in (inch), Mpc (Mega parsec), pc (parsec), lyr (light-year), au (astronomical unit), km (kilometer), mi (international mile), nmi (nautical mile), miUS (US statute mile), chain (chain), rd (rod), fath (fathom), ftUS (survey foot), Mil (Mil), μ (micron), Å (Angstrom), fermi (fermi)

### AREA

m^2 (square meter), cm^2 (square centimeter), b (barn), yd^2 (square yard), ft^2 (square feet), in^2 (square inch), km^2 (square kilometer), ha (hectare), a (are), mi^2 (square mile), miUS^2 (square statute mile), acre (acre)

### VOLUME

m^3 (cubic meter), st (stere), cm^3 (cubic centimeter), yd^3 (cubic yard), ft^3 (cubic feet), in^3 (cubic inch), l (liter), galUK (UK gallon), galC (Canadian gallon), gal (US gallon), qt (quart), pt (pint), ml (mililiter), cu (US cup), ozfl (US fluid ounce), ozUK (UK fluid ounce), tbsp (tablespoon), tsp (teaspoon), bbl (barrel), bu (bushel), pk (peck), fbm (board foot)

### TIME

yr (year), d (day), h (hour), min (minute), s (second), Hz (hertz)

### SPEED

m/s (meter per second), cm/s (centimeter per second), ft/s (feet per second), kph (kilometer per hour), mph (mile per hour), knot (nautical miles per hour), c (speed of light), ga (acceleration of gravity )

### MASS

kg (kilogram), g (gram), Lb (avoirdupois pound), oz (ounce), slug (slug), lbt (Troy pound), ton (short ton), tonUK (long ton), t (metric ton), ozt (Troy ounce), ct (carat), grain (grain), u (unified atomic mass), mol (mole)

### FORCE

N (newton), dyn (dyne), gf (gram-force), kip (kilopound-force), lbf (pound-force), pdl (poundal)

### ENERGY

J (joule), erg (erg), Kcal (kilocalorie), Cal (calorie), Btu (International table btu ), ft×lbf (foot-pound), therm (EEC therm), MeV (mega electron-volt), eV (electron-volt)

***POWER***

W (watt), hp (horse power),

***PRESSURE***

Pa (pascal), atm (atmosphere), bar (bar), psi (pounds per square inch), torr (torr), mmHg (millimeters of mercury), inHg (inches of mercury), inH20 (inches of water),

***TEMPERATURE***

° C (degree Celsius), ° F (degree Fahrenheit), K (Kelvin), ° R (degree Rankine),

***ELECTRIC CURRENT (Electric measurements)***

V (volt), A (ampere), C (coulomb), Ω (ohm), F (farad), W (watt), Fdy (faraday), H (henry), mho (mho), S (siemens), T (tesla), Wb (weber )

***ANGLE (planar and solid angle measurements)***

° (sexagesimal degree), r (radian), grad (grade), arcmin (minute of arc), arcs (second of arc), sr (steradian)

***LIGHT (Illumination measurements)***

fc (footcandle), flam (footlambert), lx (lux), ph (phot), sb (stilb), lm (lumem), cd (candela), lam (lambert)

***RADIATION***

Gy (gray), rad (rad), rem (rem), Sv (sievert), Bq (becquerel), Ci (curie), R (roentgen)

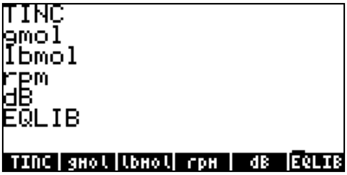
***VISCOSITY***

P (poise), St (stokes)

**Units not listed**

Units not listed in the Units menu, but available in the calculator include: gmol (gram-mole), lbmol (pound-mole), rpm (revolutions per minute), dB (decibels). These units are accessible through menu 117.02, triggered by using

MENU(117.02) in ALG mode, or 117.02 ENTER MENU in RPN mode. The menu will show in the screen as follows (use ↱▼ to show labels in display):



```
TINC
gmol
lbmol
rpm
dB
EQLIB
TINC | gmol | lbmol | rpm | dB | EQLIB
```

These units are also accessible through the catalog, for example:

| | |
|---|---|
| gmol: | ↱ _CAT ALPHA ↰ G |
| lbmol: | ↱ _CAT ALPHA ↰ L |
| rpm: | ↱ _CAT ALPHA ↰ R |
| dB: | ↱ _CAT ALPHA ↰ D |

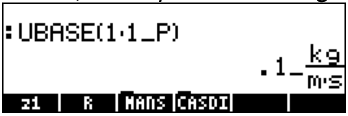
## Converting to base units

To convert any of these units to the default units in the SI system, use the function UBASE. For example, to find out what is the value of *1 poise* (unit of viscosity) in the SI units, use the following:

In ALG mode, system flag 117 set to *CHOOSE boxes*:

| | |
|---|---|
| ↱ _UNITS | Select the UNITS menu |
| OK | Select the TOOLS menu |
| ▼ OK | Select the UBASE function |
| 1 ↱ _ | Enter 1 and underline |
| ↱ _UNITS | Select the UNITS menu |
| ▲ OK | Select the VISCOSITY option |
| OK | Select the UNITS menu |
| ENTER | Convert the units |

This results in the following screen (i.e., *1 poise = 0.1 kg/(m·s)*):



```
:UBASE(1·1_P)
                    .1_kg/(m·s)
z1 | R | MANS | CASDI |  |
```

In RPN mode, system flag 117 set to *CHOOSE boxes*:

| | |
|---|---|
| 1 | Enter 1 (no underline) |
| ↱ UNITS | Select the UNITS menu |
| △ OK | Select the VISCOSITY option |
| OK | Select the unit P (poise) |
| ↱ UNITS | Select the UNITS menu |
| OK | Select the TOOLS menu |
| ▽ OK | Select the UBASE function |

In ALG mode, system flag 117 set to *SOFT menus*:

| | |
|---|---|
| ↱ UNITS | Select the UNITS menu |
| TOOLS | Select the TOOLS menu |
| UBASE | Select the UBASE function |
| 1 ↱ _ | Enter 1 and underline |
| ↱ UNITS | Select the UNITS menu |
| ↰ PREV VISC | Select the VISCOSITY option |
| P | Select the unit P (poise) |
| ENTER | Convert the units |

In RPN mode, system flag 117 set to *SOFT menus*:

| | |
|---|---|
| 1 | Enter 1 (no underline) |
| ↱ UNITS | Select the UNITS menu |
| ↰ PREV VISC | Select the VISCOSITY option |
| P | Select the unit P (poise) |
| ↱ UNITS | Select the UNITS menu |
| TOOLS | Select the TOOLS menu |
| UBASE | Select the UBASE function |

## Attaching units to numbers

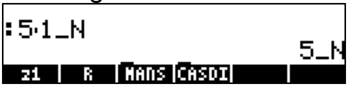
To attach a unit object to a number, the number must be followed by an underscore (↱ _ , key(8,5)). Thus, a force of 5 N will be entered as 5_N.

Here is the sequence of steps to enter this number in ALG mode, system flag 117 set to *CHOOSE boxes*:

| | |
|---|---|
| 5 ↱ _ | Enter number and underscore |
| ↱ UNITS | Access the UNITS menu |

| | |
|---|---|
| 8 OK | Select units of force (*8. Force..*) |
| OK | Select Newtons (*N*) |
| ENTER | Enter quantity with units in the stack |

The screen will look like the following:



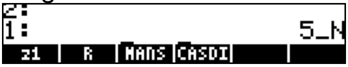
```
:5·1_N
                                5_N
```

**Note**: If you forget the underscore, the result is the expression 5*N, where N here represents a possible variable name and not Newtons.

To enter this same quantity, with the calculator in RPN mode, use the following keystrokes:

| | |
|---|---|
| 5 | Enter number (do not enter underscore) |
| ↱ UNITS | Access the UNITS menu |
| 8 OK | Select units of force (*8. Force..*) |
| OK | Select Newtons (*N*) |

Notice that the underscore is entered automatically when the RPN mode is active. The result is the following screen:



```
2:
1:                              5_N
```

As indicated earlier, if system flag 117 is set to *SOFT menus*, then the UNITS menu will show up as labels for the soft menu keys. This set up is very convenient for extensive operations with units.

The keystroke sequences to enter units when the *SOFT menu* option is selected, in both ALG and RPN modes, are illustrated next. For example, in ALG mode, to enter the quantity 5_N use:

| | |
|---|---|
| 5 ↱ _ | Enter number and underscore |
| ↱ UNITS | Access the UNITS menu |
| NXT FORCE | Select units of force |
| N | Select Newtons (*N*) |
| ENTER | Enter quantity with units in the stack |

The same quantity, entered in RPN mode uses the following keystrokes:

| | |
|---|---|
| 5 | Enter number (no underscore) |
| ↱ UNITS | Access the UNITS menu |
| NXT FORCE | Select units of force |
| N | Select Newtons (*N*) |

**Note:** You can enter a quantity with units by typing the underline and units with the ALPHA keyboard, e.g., 5 ↱ _ ALPHA N will produce the entry: 5_N

**Unit prefixes**

You can enter prefixes for units according to the following table of prefixes from the SI system.

The prefix abbreviation is shown first, followed by its name, and by the exponent x in the factor $10^x$ corresponding to each prefix:

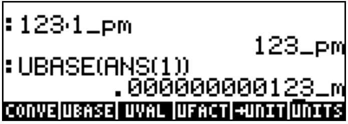
| Prefix | Name | x | Prefix | Name | x |
|---|---|---|---|---|---|
| Y | yotta | +24 | d | deci | -1 |
| Z | zetta | +21 | c | centi | -2 |
| E | exa | +18 | m | milli | -3 |
| P | peta | +15 | μ | micro | -6 |
| T | tera | +12 | n | nano | -9 |
| G | giga | +9 | p | pico | -12 |
| M | mega | +6 | f | femto | -15 |
| k,K | kilo | +3 | a | atto | -18 |
| h,H | hecto | +2 | z | zepto | -21 |
| D(*) | deka | +1 | y | yocto | -24 |

(*) In the SI system, this prefix is *da* rather than *D*. Use D for deka in the calculator, however.

To enter these prefixes, simply type the prefix using the ALPHA keyboard. For example, to enter 123 pm (1 picometer), use:

1 2 3 ↱ _ ALPHA ↰ P ALPHA ↰ M

Using UBASE to convert to the default unit (1 m) results in:

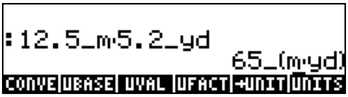


```
:123·1_pm
                    123_pm
:UBASE(ANS(1))
           .000000000123_m
```

## Operations with units

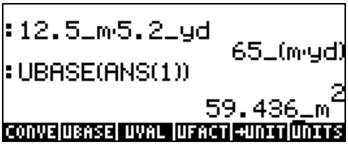
Once a quantity accompanied with units is entered into the stack, it can be used in operations similar to plain numbers, except that quantities with units cannot be used as arguments of functions (say, SQ or SIN). Thus, attempting to calculate LN(10_m) will produce an error message: *Error: Bad Argument Type.*

Here are some calculation examples using the ALG operating mode. Be warned that, when multiplying or dividing quantities with units, you must enclosed each quantity with its units between parentheses. Thus, to enter, for example, the product 12.5m × 5.2_yd, type it to read (12.5_m)*(5.2_yd) ENTER :



```
:12.5_m·5.2_yd
                65_(m·yd)
```

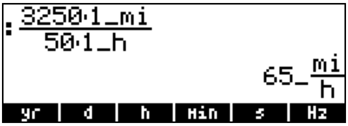
which shows as 65_(m·yd). To convert to units of the SI system, use function UBASE:



```
:12.5_m·5.2_yd
                65_(m·yd)
:UBASE(ANS(1))
                59.436_m^2
```

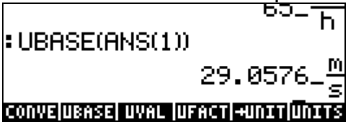
**Note:** Recall that the ANS(1) variable is available through the keystroke combination ← ANS (associated with the ENTER key).

To calculate a division, say, 3250 mi / 50 h, enter it as (3250_mi)/(50_h) ENTER :



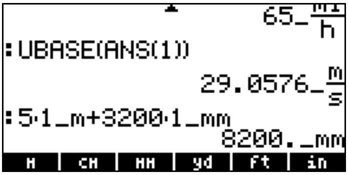
```
:3250·1_mi
  50·1_h
                      65_mi/h
```

which transformed to SI units, with function UBASE, produces:



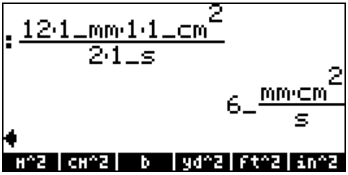
```
                      65_mi/h
:UBASE(ANS(1))
                 29.0576_m/s
```

Addition and subtraction can be performed, in ALG mode, without using parentheses, e.g., 5 m + 3200 mm, can be entered simply as 5_m + 3200_mm ENTER :



```
                      65_mi/h
:UBASE(ANS(1))
                 29.0576_m/s
:5·1_m+3200·1_mm
                    8200._mm
```

More complicated expression require the use of parentheses, e.g., (12_mm)*(1_cm^2)/(2_s) ENTER :

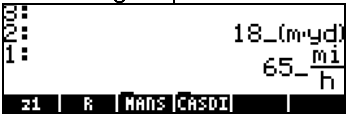


```
:12·1_mm·1·1_cm^2
      2·1_s
                 6_mm·cm^2/s
```

Stack calculations in the RPN mode, do not require you to enclose the different terms in parentheses, e.g.,

12_m ENTER 1.5_yd ENTER ×
3250_mi ENTER 50_h ENTER ÷

These operations produce the following output:

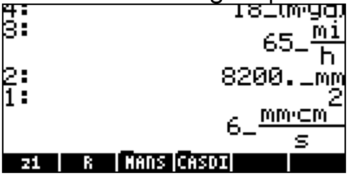


```
3:
2:                18_(m·yd)
1:                  65_mi/h
```

Also, try the following operations:

5_m ENTER 3200_mm ENTER +
12_mm ENTER 1_cm^2 ENTER × 2_s ENTER ÷

These last two operations produce the following output:



```
4:                18_(m·yd
3:                     mi
                   65_ ——
                        h
2:              8200._mm
1:                       2
                   mm·cm
                6_ ———————
                      s
```

**Note:** Units are not allowed in expressions entered in the equation writer.

## Units manipulation tools

The UNITS menu contains a TOOLS sub-menu, which provides the following functions:

| | |
|---|---|
| CONVERT(x,y): | convert unit object x to units of object y |
| UBASE(x): | convert unit object x to SI units |
| UVAL(x): | extract the value from unit object x |
| UFACT(x,y): | factors a unit x from unit object x |
| →UNIT(x,y): | combines value of x with units of y |

The UBASE function was discussed in detail in an earlier section in this chapter. To access any of these functions follow the examples provided earlier for UBASE. Notice that, while function UVAL requires only one argument, functions CONVERT, UFACT, and →UNIT require two arguments.

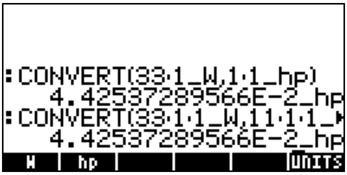
Try the following exercises, in your favorite calculator settings. The output shown below was developed in ALG mode with system flat 117 set to *SOFT menu*:

### Examples of CONVERT

These examples produce the same result, i.e., to convert 33 watts to btu's

CONVERT(33_W,1_hp) ENTER
CONVERT(33_W,11_hp) ENTER

These operations are shown in the screen as:

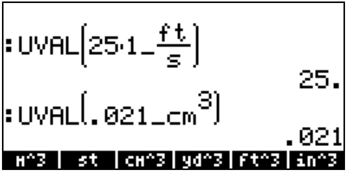


```
:CONVERT(33·1_W,1·1_hp)
   4.42537289566E-2_hp
:CONVERT(33·1·1_W,11·1·1_▸
   4.42537289566E-2_hp
```

**Examples of UVAL:**

UVAL(25_ft/s) ENTER

UVAL(0.021_cm^3) ENTER

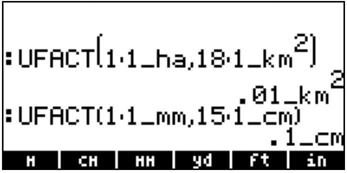


```
:UVAL(25·1_ft/s)
                      25.
:UVAL(.021_cm^3)
                     .021
```

**Examples of UFACT**

UFACT(1_ha,18_km^2) ENTER

UFACT(1_mm,15.1_cm) ENTER

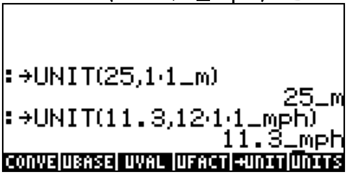


```
:UFACT(1·1_ha,18·1_km^2)
                  .01_km^2
:UFACT(1·1_mm,15·1_cm)
                     .1_cm
```

**Examples of →UNIT**

→UNIT(25,1_m) ENTER

→UNIT(11.3,1_mph) ENTER



```
:→UNIT(25,1·1_m)
                    25_m
:→UNIT(11.3,12·1·1_mph)
                11.3_mph
```

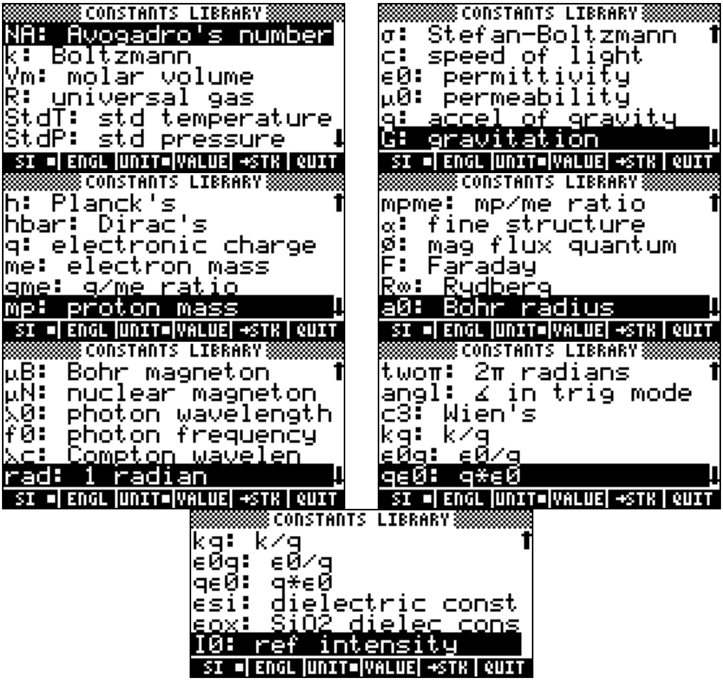
## Physical constants in the calculator

Following along the treatment of units, we discuss the use of physical constants that are available in the calculator's memory. These physical constants are contained in a *constants library* activated with the command CONLIB. To launch this command you could simply type it in the stack:

ALPHA ALPHA C O N L I B ALPHA ENTER

or, you can select the command CONLIB from the command catalog, as follows: First, launch the catalog by using: ‍→ CAT ALPHA C . Next, use the up and down arrow keys ▲▼ to select CONLIB. Finally, press the F6 (OK) soft menu key. Press ENTER, if needed.

The constants library screen will look like the following (use the down arrow key to navigate through the library):



```
CONSTANTS LIBRARY
NA: Avogadro's number
k: Boltzmann
Vm: molar volume
R: universal gas
StdT: std temperature
StdP: std pressure
SI ■ ENGL UNIT■ VALUE →STK QUIT
```

```
CONSTANTS LIBRARY
σ: Stefan-Boltzmann
c: speed of light
ε0: permittivity
μ0: permeability
g: accel of gravity
G: gravitation
SI ■ ENGL UNIT■ VALUE →STK QUIT
```

```
CONSTANTS LIBRARY
h: Planck's
hbar: Dirac's
q: electronic charge
me: electron mass
qme: q/me ratio
mp: proton mass
SI ■ ENGL UNIT■ VALUE →STK QUIT
```

```
CONSTANTS LIBRARY
mpme: mp/me ratio
α: fine structure
Ø: mag flux quantum
F: Faraday
R∞: Rydberg
a0: Bohr radius
SI ■ ENGL UNIT■ VALUE →STK QUIT
```

```
CONSTANTS LIBRARY
μB: Bohr magneton
μN: nuclear magneton
λ0: photon wavelength
f0: photon frequency
λc: Compton wavelen
rad: 1 radian
SI ■ ENGL UNIT■ VALUE →STK QUIT
```

```
CONSTANTS LIBRARY
twoπ: 2π radians
angl: ∡ in trig mode
c3: Wien's
kq: k/q
ε0q: ε0/q
qε0: q*ε0
SI ■ ENGL UNIT■ VALUE →STK QUIT
```

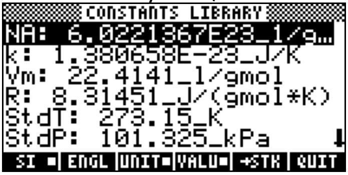
```
CONSTANTS LIBRARY
kq: k/q
ε0q: ε0/q
qε0: q*ε0
εsi: dielectric const
εox: SiO2 dielec cons
I0: ref intensity
SI ■ ENGL UNIT■ VALUE →STK QUIT
```

The soft menu keys corresponding to this CONSTANTS LIBRARY screen include the following functions:

| | |
|---|---|
| SI | when selected, constants values are shown in SI units |
| ENGL | when selected, constants values are shown in English units (*) |
| UNIT | when selected, constants are shown with units attached (*) |

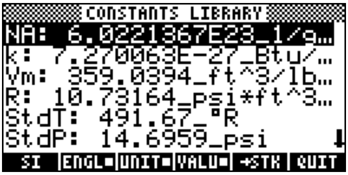
VALUE when selected, constants are shown without units
→STK copies value (with or without units) to the stack
QUIT exit constants library
(*) Active only if the function VALUE is active.

This is the way the top of the CONSTANTS LIBRARY screen looks when the option VALUE is selected (units in the SI system):



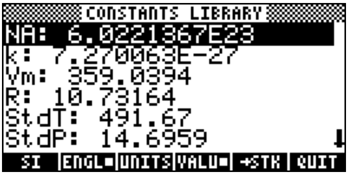
```
CONSTANTS LIBRARY
NA: 6.0221367E23_1/g…
k: 1.380658E-23_J/K
Vm: 22.4141_l/gmol
R: 8.31451_J/(gmol*K)
StdT: 273.15_K
StdP: 101.325_kPa
```

To see the values of the constants in the English (or Imperial) system, press the ENGL option:



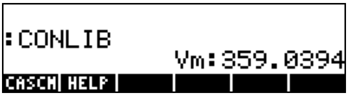
```
CONSTANTS LIBRARY
NA: 6.0221367E23_1/g…
k: 7.270063E-27_Btu/…
Vm: 359.0394_ft^3/lb…
R: 10.73164_psi*ft^3…
StdT: 491.67_°R
StdP: 14.6959_psi
```

If we de-select the UNITS option (press UNITS ) only the values are shown (English units selected in this case):



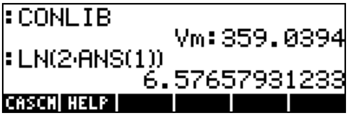
```
CONSTANTS LIBRARY
NA: 6.0221367E23
k: 7.270063E-27
Vm: 359.0394
R: 10.73164
StdT: 491.67
StdP: 14.6959
```

To copy the value of Vm to the stack, select the variable name, and press !→STK, then, press QUIT. For the calculator set to the ALG, the screen will look like this:



```
:CONLIB
                Vm:359.0394
```

The display shows what is called a *tagged value*, `Vm:359.0394`. In here, Vm, is the *tag* of this result. Any arithmetic operation with this number will ignore the tag. Try, for example: ↱ LN 2 × ↰ ANS ENTER, which produces:

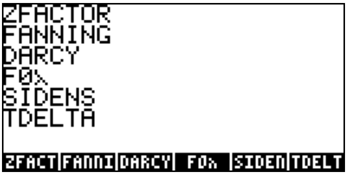


```
:CONLIB
                 Vm:359.0394
:LN(2·ANS(1))
              6.57657931233
```

The same operation in RPN mode will require the following keystrokes (after the value of Vm was extracted from the constants library): 2 ENTER × ↱ LN

## Special physical functions

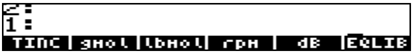
Menu 117, triggered by using MENU(117) in ALG mode, or 117 ENTER MENU in RPN mode, produces the following menu (labels listed in the display by using ↱ ▼):



```
ZFACTOR
FANNING
DARCY
F0λ
SIDENS
TDELTA
```

The functions include:
ZFACTOR: gas compressibility Z factor function
FANNING: Fanning friction factor for fluid flow
DARCY: Darcy-Weisbach friction factor for fluid flow
F0λ: Black body emissive power function
SIDENS: Silicon intrinsic density
TDELTA: Temperature delta function

In the second page of this menu (press NXT) we find the following items:



```
2:
1:
```

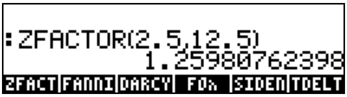
In this menu page, there is one function (TINC) and a number of units described in an earlier section on units (see above). The function of interest is:

TINC: temperature increment command

Out of all the functions available in this MENU (UTILITY menu), namely, ZFACTOR, FANNING, DARCY, F0λ, SIDENS, TDELTA, and TINC, functions FANNING and DARCY are described in Chapter 6 in the context of solving equations for pipeline flow. The remaining functions are described following.

## Function ZFACTOR

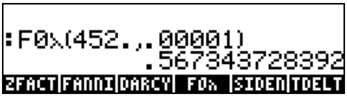
Function ZFACTOR calculates the gas compressibility correction factor for nonideal behavior of hydrocarbon gas. The function is called by using ZFACTOR($x_T$, $y_P$), where $x_T$ is the reduced temperature, i.e., the ratio of actual temperature to pseudo-critical temperature, and $y_P$ is the reduced pressure, i.e., the ratio of the actual pressure to the pseudo-critical pressure. The value of $x_T$ must be between 1.05 and 3.0, while the value of $y_P$ must be between 0 and 30. Example, in ALG mode:



```
:ZFACTOR(2.5,12.5)
          1.25980762398
```

## Function F0λ

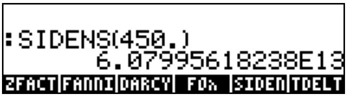
Function F0λ (T, λ) calculates the fraction (dimensionless) of total black-body emissive power at temperature T between wavelengths 0 and λ. If no units are attached to T and λ, it is implied that T is in K and λ in m. Example, in ALG mode:



```
:F0λ(452.,.00001)
           .567343728392
```

## Function SIDENS

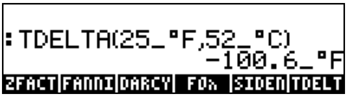
Function SIDENS(T) calculates the intrinsic density of silicon (in units of 1/cm$^3$) as a function of temperature T (T in K), for T between 0 and 1685 K. For example,



```
:SIDENS(450.)
       6.07995618238E13
```

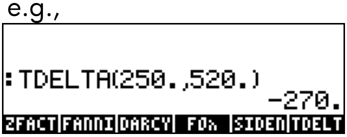
## Function TDELTA

Function TDELTA($T_0$,$T_f$) yields the temperature increment $T_f - T_0$. The result is returned with the same units as $T_0$, if any. Otherwise, it returns simply the difference in numbers. For example,



```
:TDELTA(25_°F,52_°C)
                -100.6_°F
```

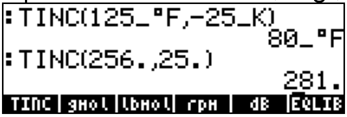
The purpose of this function is to facilitate the calculation of temperature differences given temperatures in different units. Otherwise, it's simply calculates a subtraction, e.g.,



```
:TDELTA(250.,520.)
                    -270.
```

## Function TINC

Function TINC($T_0$,$\Delta T$) calculates $T_0$+DT. The operation of this function is similar to that of function TDELTA in the sense that it returns a result in the units of $T_0$. Otherwise, it returns a simple addition of values, e.g.,



```
:TINC(125_°F,-25_K)
                   80_°F
:TINC(256.,25.)
                    281.
```

# Defining and using functions

Users can define their own functions by using the DEF command available thought the keystroke sequence ↰ DEF (associated with the 2 key). The function must be entered in the following format:

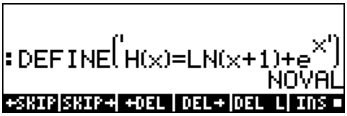
*Function_name(arguments) = expression_containing_arguments*

For example, we could define a simple function*H(x) = ln(x+1) + exp(-x).*

Suppose that you have a need to evaluate this function for a number of discrete values and, therefore, you want to be able to press a single button and get the result you want without having to type the expression in the right-hand side for each separate value. In the following example, we assume you have set your calculator to ALG mode. Enter the following sequence of keystrokes:

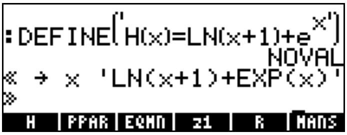
← DEF ' ALPHA H ← () ALPHA ← X ▶ → =
→ LN ALPHA ← X + 1 ▶ + ← e^x ALPHA ← X ENTER

The screen will look like this:



```
:DEFINE('H(x)=LN(x+1)+e^x')
                      NOVAL
```

Press the VAR key, and you will notice that there is a new variable in your soft menu key (H). To see the contents of this variable press → H. The screen will show now:



```
:DEFINE('H(x)=LN(x+1)+e^x')
                      NOVAL
« → x 'LN(x+1)+EXP(x)'
»
```

Thus, the variable H contains a program defined by:

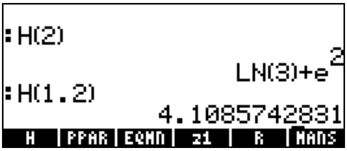
<< → x 'LN(x+1) + EXP(x)' >>

This is a simple program in the default programming language of the HP 48 G series, and also incorporated in the HP 49 G series. This programming language is called UserRPL. The program shown above is relatively simple and consists of two parts, contained between the program containers « »:

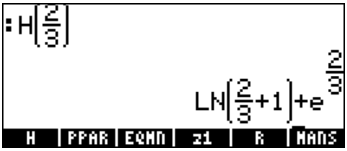
- Input: → x → x
- Process: 'LN(x+1) + EXP(x) '

This is to be interpreted as saying: enter a value that is temporarily assigned to the name *x* (referred to as a local variable), evaluate the expression between quotes that contain that local variable, and show the evaluated expression.

To activate the function in ALG mode, type the name of the function followed by the argument between parentheses, e.g., H ← () 2 ENTER. Some examples are shown below:



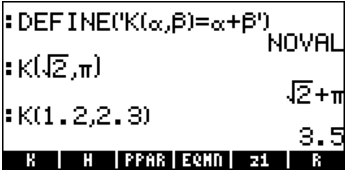
```
:H(2)
                LN(3)+e^2
:H(1.2)
             4.1085742831
 H  PPAR EQNN  z1   R  MANS
```



```
:H(2/3)
         LN(2/3+1)+e^(2/3)
 H  PPAR EQNN  z1   R  MANS
```

In the RPN mode, to activate the function enter the argument first, then press the soft menu key corresponding to the variable name ▦. For example, you could try: 2 ENTER ▦ . The other examples shown above can be entered by using: / • 2 ENTER ▦ , 2 ÷ 3 ENTER ▦ .

Functions can have more than 2 arguments. For example, the screen below shows the definition of the function *K(α,β) = α+β,* and its evaluation with the arguments *K(√2,π),* and *K(1.2,2.3):*



```
:DEFINE('K(α,β)=α+β')
                    NOVAL
:K(√2,π)
                     √2+π
:K(1.2,2.3)
                      3.5
 K   H  PPAR EQNN  z1   R
```

The contents of the variable K are: << → α β 'α+β' >>.

## Functions defined by more than one expression

In this section we discuss the treatment of functions that are defined by two or more expressions. An example of such functions would be

$$f(x) = \begin{Bmatrix} 2 \cdot x - 1, & x < 0 \\ x^2 - 1, & x > 0 \end{Bmatrix}$$

The HP 49 G provides the function IFTE (IF-Then-Else) to describe such functions.

### The IFTE function

The IFTE function is written as IFTE(*condition*, *operation_if_true*, *operation_if_false*) If *condition* is true then *operation_if_true* is performed, else *operation_if_false* is performed. For example, we can write 'f(x) = IFTE(x>0, x^2-1, 2*x-1)', to describe the function listed above. Function IFTE is accessible from the function catalog (↱ CAT ). The symbol '>' (greater than) is available as

(associated with the 1/x key). To define this function in ALG mode use the command: DEF(f(x) = IFTE(x>0, x^2-1, 2*x-1))
then, press ENTER. In RPN mode, type the function definition between apostrophes:

‘f(x) = IFTE(x>0, x^2-1, 2*x-1)’

then press ↰ DEF .

Press VAR to recover your variable menu. The function f should be available in your soft key menu. Press ↱ f to see the resulting program:

<< → x ‘IFTE(x>0, x^2-1, 2*x-1)’ >>

To evaluate the function in ALG mode, type the function name, f, followed by the number at which you want to evaluate the function, e.g., f(2), then press ENTER. In RPN mode, enter a number and press f. Check, for example, that *f(2) = 3*, while *f(-2) = -5*.

**Combined IFTE functions**

To program a more complicated function such as

$$g(x)=\begin{cases} -x, & x<-2 \\ x+1, & -2\le x<0 \\ x-1, & 0\le x<2 \\ x^2, & x\ge 2 \end{cases}$$

you can combine several levels of the IFTE function, i.e.,

‘g(x) = IFTE(x<-2, -x, IFTE(x<0, x+1, IFTE(x<2, x-1, x^2)))’,

Define this function by any of the means presented above, and check that g(-3) = 3, g(-1) = 0, g(1) = 0, g(3) = 9.

# Chapter 4
# Calculations with complex numbers

This chapter shows examples of calculations and application of functions to complex numbers.

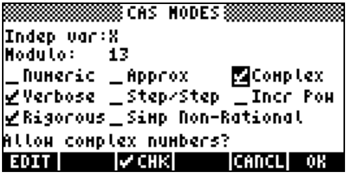
## Definitions

A *complex number z* is a number written as $z = x + iy$, where $x$ and $y$ are real numbers, and $i$ is the *imaginary unit* defined by $i^2 = -1$. The complex number $x+iy$ has a *real part*, $x = Re(z)$, and an *imaginary part*, $y = Im(z)$. We can think of a complex number as a point $P(x,y)$ in the x-y plane, with the x-axis referred to as the *real axis*, and the y-axis referred to as the *imaginary axis*. Thus, a complex number represented in the form $x+iy$ is said to be in its *Cartesian representation*. An alternative Cartesian representation is the ordered pair $z = (x,y)$. A complex number can also be represented in polar coordinates (*polar representation*) as $z = re^{i\theta} = r\cdot\cos\theta + i\, r\cdot\sin\theta$, where $r = |z| = \sqrt{x^2 + y^2}$ is the *magnitude* of the complex number z, and $\theta = Arg(z) = arctan(y/x)$ is the *argument* of the complex number z. The relationship between the Cartesian and polar representation of complex numbers is given by the *Euler formula*: $e^{i\theta} = \cos\theta + i\sin\theta$. The *complex conjugate* of a complex number $z = x + iy = re^{i\theta}$, is $\bar{z} = x - iy = re^{-i\theta}$. The complex conjugate of $i$ can be thought of as the reflection of z about the real ($x$) axis. Similarly, the *negative* of z, $-z = -x-iy = -re^{i\theta}$, can be thought of as the reflection of $z$ about the origin.

## Setting the calculator to COMPLEX mode

When working with complex numbers it is a good idea to set the calculator to complex mode, use the following keystrokes: MODE CAS 2 ▼ ▼ ▶ CHK

The COMPLEX mode will be selected if the CAS MODES screen shows the option *_Complex* checked off, i.e.,



```
CAS MODES
Indep var:X
Modulo:   13
_Numeric _Approx    ☑Complex
✓Verbose _Step/Step _Incr Pow
✓Rigorous_Simp Non-Rational
Allow complex numbers?
EDIT      ✓CHK      CANCL  OK
```

Press OK, twice, to return to the stack.

## Entering complex numbers

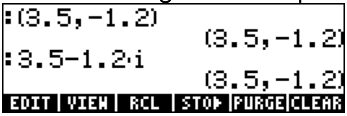
Complex numbers in the calculator can be entered in either of the two Cartesian representations, namely, *x+iy*, or *(x,y)*. The results in the calculator will be shown in the ordered-pair format, i.e., *(x,y)*. For example, with the calculator in ALG mode, the complex number *(3.5,-1.2),* is entered as:

← () 3 • 5 ↱ , +/- 1 • 2 ENTER

A complex number can also be entered in the form *x+iy*. For example, in ALG mode, *3.5-1.2i* is entered as:

3 • 5 − 1 • 2 × ← i ENTER

The following screen results after entering these complex numbers:



```
:(3.5,-1.2)
                 (3.5,-1.2)
:3.5-1.2·i
                 (3.5,-1.2)
EDIT VIEW RCL STO▸ PURGE CLEAR
```

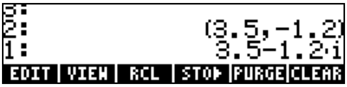
In RPN mode, these numbers will be entered using the following keystrokes:

← () 3 • 5 ↱ , 1 • 2 +/- ENTER

(Notice that the change-sign keystroke is entered after the number 1.2 has been entered, in the opposite order as the ALG mode exercise), and

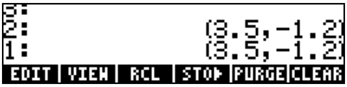
' 3 • 5 − 1 • 2 × ↱ i ENTER

(Notice the need to enter an apostrophe before typing the number 3.5-1.2i in RPN mode). The resulting RPN screen will be:



```
3:
2:               (3.5,-1.2)
1:                3.5-1.2·i
EDIT VIEW RCL STO▸ PURGE CLEAR
```

Notice that the last entry shows a complex number in the form *x+iy*. This is so because the number was entered between apostrophes, which represents an algebraic expression. To evaluate this number use the EVAL key( EVAL ).

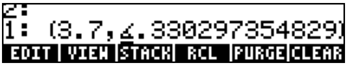


```
3:
2:               (3.5,-1.2)
1:               (3.5,-1.2)
EDIT VIEW RCL STO▸ PURGE CLEAR
```

Once the algebraic expression is evaluated, you recover the complex number (3.5,1.2).

## Polar representation of a complex number

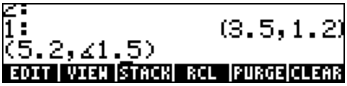
The result shown above represents a Cartesian (rectangular) representation of the complex number 3.5-1.2i. A polar representation is possible if we change the coordinate system to cylindrical or polar, by using function CYLIN. You can find this function in the catalog (↱ CAT). Changing to polar shows the result:



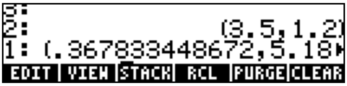
```
2:
1: (3.7,∡.330297354829)
```

For this result the angular measure is set to radians (you can always change to radians by using function RAD). The result shown above represents a magnitude, 3.7, and an angle 0.33029…. The angle symbol (∠) is shown in front of the angle measure.

Return to Cartesian or rectangular coordinates by using function RECT (available in the catalog, ↱ CAT). A complex number in polar representation is written as $z = r\cdot e^{i\theta}$. You can enter this complex number into the calculator by using an ordered pair of the form (r, ∠θ). The angle symbol (∠) can be entered as ALPHA ↱ 6. For example, the complex number $z = 5.2e^{1.5i}$, can be entered as follows (the figures show the stack, before and after entering the number):



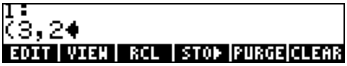
```
2:
1:            (3.5,1.2)
(5.2,∡1.5)
```



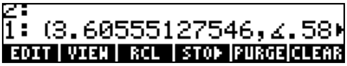
```
3:
2:            (3.5,1.2)
1: (.367833448672,5.18▸
```

Because the coordinate system is set to rectangular (or Cartesian), the calculator automatically converts the number entered to Cartesian coordinates, i.e., $x = r \cos \theta$, $y = r \sin \theta$, resulting, for this case, in (0.3678…, 5.18…).

On the other hand, if the coordinate system is set to cylindrical coordinates (use CYLIN), entering a complex number (x,y), where x and y are real numbers, will produce a polar representation. For example, in cylindrical coordinates, enter the number (3.,2.). The figure below shows the RPN stack, before and after entering this number:



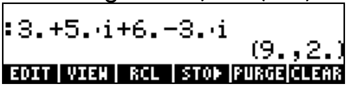
```
1:
(3,2◂
```



```
2:
1: (3.60555127546,∡.58▸
```

## Simple operations with complex numbers

Complex numbers can be combined using the four fundamental operations (+ − × ÷). The results follow the rules of algebra with the caveat that $i^2 = -1$. Operations with complex numbers are similar to those with real numbers. For example, with the calculator in ALG mode and the CAS set to *Complex*, we'll attempt the following sum: (3+5i) + (6-3i):



```
:3.+5.·i+6.-3.·i
                  (9.,2.)
```

Notice that the real parts (3+6) and imaginary parts (5-3) are combined together and the result given as an ordered pair with real part 9 and imaginary part 2. Try the following operations on your own:

(5-2i) - (3+4i) = (2,-6)

(3-i)·(2-4i) = (2,-14)

(5-2i)/(3+4i) = (0.28,-1.04)

1/(3+4i) = (0.12, -0.16)

**Notes:**

The product of two numbers is represented by: $(x_1+iy_1)(x_2+iy_2) = (x_1x_2 - y_1y_2) + i\,(x_1y_2 + x_2y_1)$.

The division of two complex numbers is accomplished by multiplying both numerator and denominator by the complex conjugate of the denominator, i.e.,

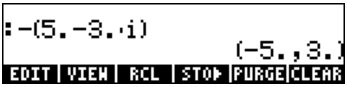
$$\frac{x_1+iy_1}{x_2+iy_2} = \frac{x_1+iy_1}{x_2+iy_2}\cdot\frac{x_2-iy_2}{x_2-iy_2} = \frac{x_1x_2+y_1y_2}{x_2^2+y_2^2} + i\cdot\frac{x_2y_1-x_1y_2}{x_2^2+y_2^2}$$

Thus, the inverse function INV (activated with the 1/x key) is defined as

$$\frac{1}{x+iy} = \frac{1}{x+iy}\cdot\frac{x-iy}{x-iy} = \frac{x}{x^2+y^2} + i\cdot\frac{y}{x^2+y^2}$$

## Changing sign of a complex number

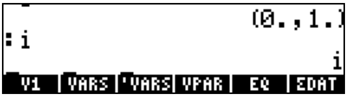
Changing the sign of a complex number can be accomplished by using the +/- key, e.g., -(5-3i) = -5 + 3i



```
:-(5.-3.·i)
                    (-5.,3.)
```

## Entering the unit imaginary number

To enter the unit imaginary number type : ↰ *i*



```
                    (0.,1.)
:i
                          i
```

Notice that the number *i* is entered as the ordered pair *(0,1)* if the CAS is set to APPROX mode. In EXACT mode, the unit imaginary number is entered as *i*.

### Other operations

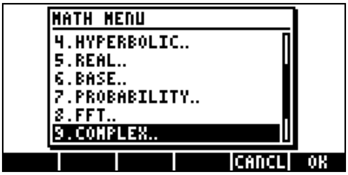
Operations such as magnitude, argument, real and imaginary parts, and complex conjugate are available through the CMPLX menus detailed later.

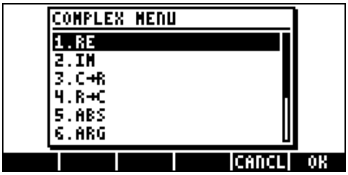
## The CMPLX menus

There are two CMPLX (CoMPLeX numbers) menus available in the calculator. One is available through the MTH menu (introduced in Chapter 3) and one directly into the keyboard (↱ CMPLX ). The two CMPLX menus are presented next.

### CMPLX menu through the MTH menu

Assuming that system flag 117 is set to **CHOOSE boxes** (see Chapter 2), the CMPLX sub-menu within the MTH menu is accessed by using: ↰ MTH 9 OK . The following sequence of screen shots illustrates these steps:

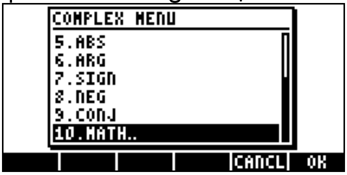




The first menu (options 1 through 6) shows the following functions:

RE(z) : Real part of a complex number
IM(z) : Imaginary part of a complex number
C→R(z) : Takes a complex number (x,y) and separates it into its real and imaginary parts
R→C(x,y): Forms the complex number *(x,y)* out of real numbers x and y
ABS(z) : Calculates the magnitude of a complex number or the absolute value of a real number.
ARG(z): Calculates the argument of a complex number.

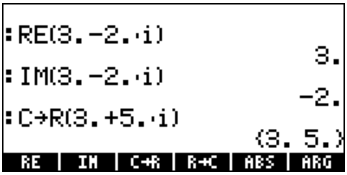
The remaining options (options 7 through 10) are the following:



SIGN(z) : Calculates a complex number of unit magnitude as z/|z|.
NEG : Changes the sign of z
CONJ(z): Produces the complex conjugate of z

Examples of applications of these functions are shown next. Recall that, for ALG mode, the function must precede the argument, while in RPN mode, you enter the argument first, and then select the function. Also, recall that you can get these functions as soft menus by changing the setting of system flag 117 (See Chapter 3).

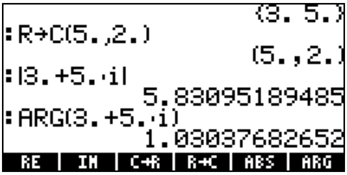
This first screen shows functions RE, IM, and C→R. Notice that the last function returns a list {3. 5.} representing the real and imaginary components of the complex number:



```
:RE(3.-2.·i)
                    3.
:IM(3.-2.·i)
                   -2.
:C→R(3.+5.·i)
               {3. 5.}
```

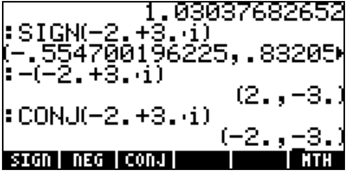
The following screen shows functions R→C, ABS, and ARG. Notice that the ABS function gets translated to |3.+5.·i|, the notation of the absolute value.

Also, the result of function ARG, which represents an angle, will be given in the units of angle measure currently selected. In this example, ARG(3.+5.·i) = 1.0303… is given in radians.



```
                    (3. 5.)
:R→C(5.,2.)
                    (5.,2.)
:|3.+5.·i|
               5.83095189485
:ARG(3.+5.·i)
               1.03037682652
```

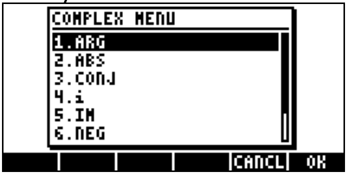
In the next screen we present examples of functions SIGN, NEG (which shows up as the negative sign - ), and CONJ.



```
               1.03037682652
:SIGN(-2.+3.·i)
(-.554700196225,.83205▸
:-(-2.+3.·i)
                   (2.,-3.)
:CONJ(-2.+3.·i)
                  (-2.,-3.)
```

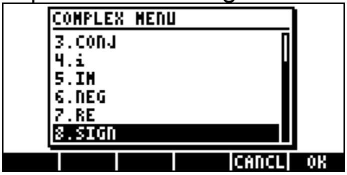
### CMPLX menu in keyboard

A second CMPLX menu is accessible by using the right-shift option associated with the / key, i.e., ↱ CMPLX . With system flag 117 set to CHOOSE boxes, the keyboard CMPLX menu shows up as the following screens:



COMPLEX MENU

1.ARG
2.ABS
3.CONJ
4.i
5.IM
6.NEG



COMPLEX MENU

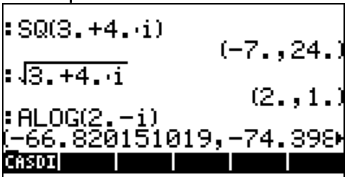
3.CONJ
4.i
5.IM
6.NEG
7.RE
8.SIGN

The resulting menu include some of the functions already introduced in the previous section, namely, ARG, ABS, CONJ, IM, NEG, RE, and SIGN. It also includes function *i* which serves the same purpose as the keystroke combination ↰ *i* , i.e., to enter the unit imaginary number *i* in an expression.

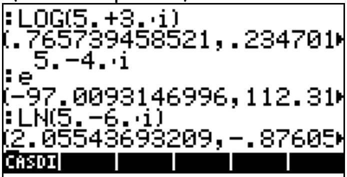
The keyboard-base CMPLX menu is an alternative to the MTH-based CMPLX menu containing the basic complex number functions. Try the examples shown earlier using the keyboard-based CMPLX menu for practice.

## Functions applied to complex numbers

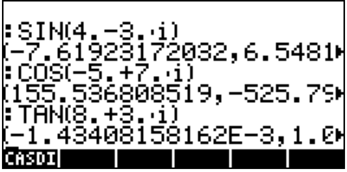
Many of the keyboard-based functions defined in Chapter 3 for real numbers, e.g., SQ, ,LN, $e^x$, LOG, $10^X$, SIN, COS, TAN, ASIN, ACOS, ATAN, can be applied to complex numbers. The result is another complex number, as illustrated in the following examples. To apply this functions use the same procedure as presented for real numbers (see Chapter 3).



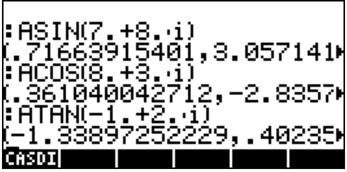
```
:SQ(3.+4.·i)
                    (-7.,24.)
:√3.+4.·i
                      (2.,1.)
:ALOG(2.-i)
(-66.820151019,-74.398▸
```



```
:LOG(5.+3.·i)
(.765739458521,.234701▸
   5.-4.·i
:e
(-97.0093146996,112.31▸
:LN(5.-6.·i)
(2.05543693209,-.87605▸
```



```
:SIN(4.-3.·i)
(-7.61923172032,6.5481▸
:COS(-5.+7.·i)
(155.536808519,-525.79▸
:TAN(8.+3.·i)
(-1.43408158162E-3,1.0▸
```

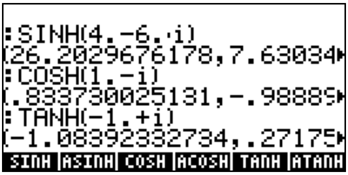


```
:ASIN(7.+8.·i)
(.71663915401,3.057141▸
:ACOS(8.+3.·i)
(.361040042712,-2.8357▸
:ATAN(-1.+2.·i)
(-1.33897252229,.40235▸
```

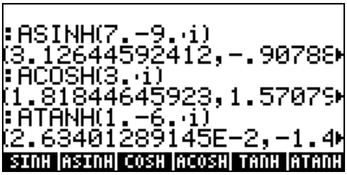
**Note:** When using trigonometric functions and their inverses with complex numbers the arguments are no longer angles. Therefore, the angular measure selected for the calculator has no bearing in the calculation of these functions with complex arguments. To understand the way that trigonometric functions, and other functions, are defined for complex numbers consult a book on complex variables.

### Functions from the MTH menu

The hyperbolic functions and their inverses, as well as the Gamma, PSI, and Psi functions (special functions) were introduced and applied to real numbers in Chapter 3. These functions can also be applied to complex numbers by following the procedures presented in Chapter 3. Some examples are shown below:

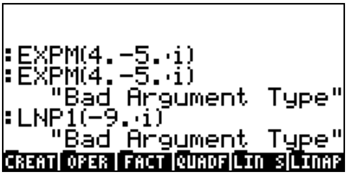


```
:SINH(4.-6.·i)
(26.2029676178,7.63034▸
:COSH(1.-i)
(.833730025131,-.98889▸
:TANH(-1.+i)
(-1.08392332734,.27175▸
```

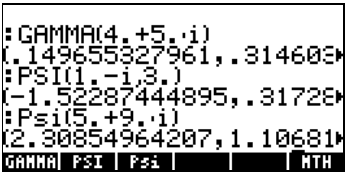


```
:ASINH(7.-9.·i)
(3.12644592412,-.90788▸
:ACOSH(3.·i)
(1.81844645923,1.57079▸
:ATANH(1.-6.·i)
(2.63401289145E-2,-1.4▸
```

The following screen shows that functions EXPM and LNP1 do not apply to complex numbers. However, functions GAMMA, PSI, and Psi accept complex numbers:



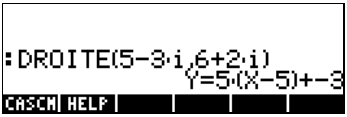
```
:EXPM(4.-5.·i)
:EXPM(4.-5.·i)
   "Bad Argument Type"
:LNP1(-9.·i)
   "Bad Argument Type"
```



```
:GAMMA(4.+5.·i)
(.149655327961,.314603▸
:PSI(1.-i,3.)
(-1.52287444895,.31728▸
:Psi(5.+9.·i)
(2.30854964207,1.10681▸
```

## Function DROITE: equation of a straight line

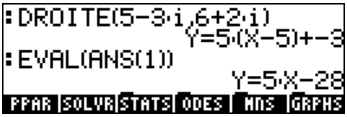
Function DROITE takes as argument two complex numbers, say, $x_1+iy_1$ and $x_2+iy_2$, and returns the equation of the straight line, say, $y = a+bx$, that contains the points $(x_1,y_1)$ and $(x_2,y_2)$. For example, the line between points A(5,-3) and B(6,2) can be found as follows (example in Algebraic mode):



```
:DROITE(5-3·i,6+2·i)
              Y=5·(X-5)+-3
```

Function DROITE is found in the command catalog (↱ CAT ).

Using EVAL(ANS(1)) simplifies the result to:



```
:DROITE(5-3·i,6+2·i)
              Y=5·(X-5)+-3
:EVAL(ANS(1))
                  Y=5·X-28
```

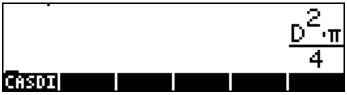
# Chapter 5
# Algebraic and arithmetic operations

An algebraic object, or simply, algebraic, is any number, variable name or algebraic expression that can be operated upon, manipulated, and combined according to the rules of algebra. Examples of algebraic objects are the following:

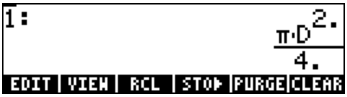
- A number: 12.3, 15.2_m, 'π', 'e', 'i'
- A variable name: 'a', 'ux', 'width', etc.
- An expression: 'p*D^2/4','f*(L/D)*(V^2/(2*g))'
- An equation: 'Q=(Cu/n)*A(y)*R(y)^(2/3)*So^0.5'

## Entering algebraic objects

Algebraic objects can be created by typing the object between single quotes directly into stack level 1 or by using the equation writer ⮣ EQW . For example, to enter the algebraic object 'π*D^2/4' directly into stack level 1 use: ' ← π × ALPHA D $y^x$ 2 ÷ 4 ENTER . The resulting screen is shown next for both the ALG mode (left-hand side) and the RPN mode (right-hand side):

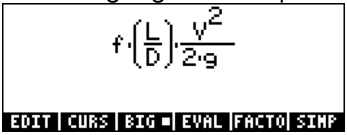


$\frac{D^2\cdot\pi}{4}$



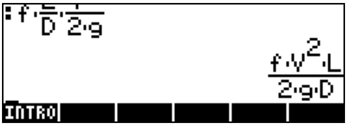
1: $\frac{\pi\cdot D^{2.}}{4.}$

An algebraic object can also be built in the Equation Writer and then sent to the stack. The operation of the Equation Writer was described in Chapter 2. As an exercise, build the following algebraic object in the Equation Writer:



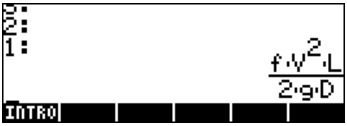
$f\cdot\left(\frac{L}{D}\right)\cdot\frac{V^2}{2\cdot g}$

After building the object, press to show it in the stack (ALG and RPN modes shown below):



:$f\cdot\frac{L}{D}\cdot\frac{1}{2\cdot g}$

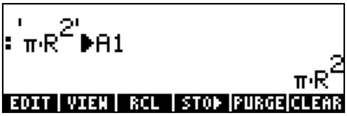
$\frac{f\cdot V^2\cdot L}{2\cdot g\cdot D}$



3:
2:
1: $\frac{f\cdot V^2\cdot L}{2\cdot g\cdot D}$

## Simple operations with algebraic objects

Algebraic objects can be added, subtracted, multiplied, divided (except by zero), raised to a power, used as arguments for a variety of standard functions (exponential, logarithmic, trigonometry, hyperbolic, etc.), as you would any real or complex number. To demonstrate basic operations with algebraic objects, let's create a couple of objects, say 'π*R^2' and 'g*t^2/4', and store them in variables A1 and A2 (See Chapter 2 to learn how to create variables and store values in them). Here are the keystrokes for storing variables A1 in ALG mode: ' ← π × ALPHA R $Y^x$ 2 ▶ STO▶ ALPHA A 1 ENTER, resulting in:

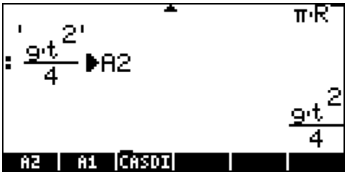


$:\ '\pi\cdot R^2' \blacktriangleright A1$

$\pi\cdot R^2$

The keystrokes corresponding to RPN mode are: ' ← π × ALPHA R $Y^x$ 2 ENTER ALPHA A 1 STO▶

After storing the variable A2 and pressing the key, the screen will show the variables as follows:



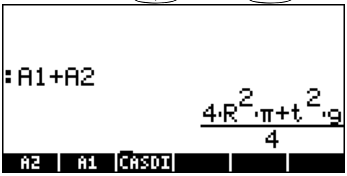
$\pi\cdot R$

$:\ '\frac{g\cdot t^2}{4}' \blacktriangleright A2$

$\frac{g\cdot t^2}{4}$

In ALG mode, the following keystrokes will show a number of operations with the algebraics contained in variables A1 and A2 (press VAR to recover variable menu):

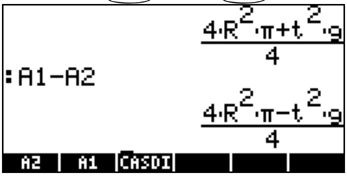
A1 + A2 ENTER



$:A1+A2$

$\frac{4\cdot R^2\cdot\pi+t^2\cdot g}{4}$

A1 − A2 ENTER

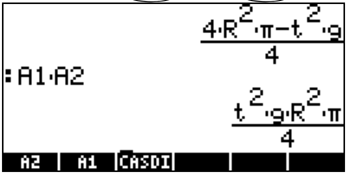


$\frac{4\cdot R^2\cdot\pi+t^2\cdot g}{4}$

$:A1-A2$

$\frac{4\cdot R^2\cdot\pi-t^2\cdot g}{4}$

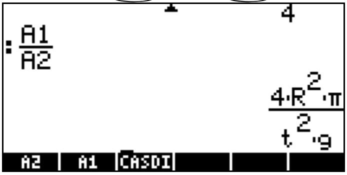
A1 × A2 ENTER


$\frac{4\cdot R^2\cdot\pi-t^2\cdot g}{4}$

:A1·A2

$\frac{t^2\cdot g\cdot R^2\cdot\pi}{4}$

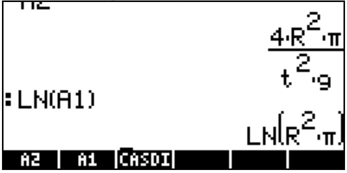
A1 ÷ A2 ENTER


4

:$\frac{A1}{A2}$

$\frac{4\cdot R^2\cdot\pi}{t^2\cdot g}$

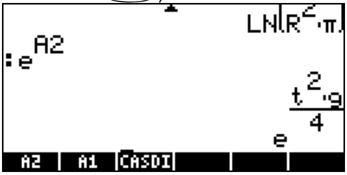
↱ LN A1


$\frac{4\cdot R^2\cdot\pi}{t^2\cdot g}$

:LN(A1)

$LN(R^2\cdot\pi)$

↰ e^x A2


$LN(R^2\cdot\pi)$

:$e^{A2}$

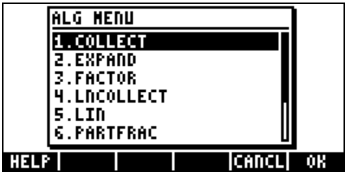
$e^{\frac{t^2\cdot g}{4}}$

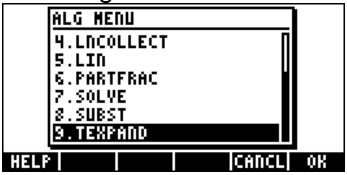
The same results are obtained in RPN mode if using the following keystrokes:

A1 ENTER A2 + A1 ENTER A2 −
A1 ENTER A2 × A1 ENTER A2 ÷
A1 ENTER ↱ LN A2 ENTER ↰ e^x

## Functions in the ALG menu

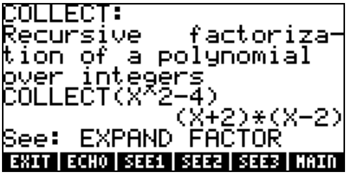
The ALG (Algebraic) menu is available by using the keystroke sequence ↱ ALG (associated with the ↱ key). With system flag 117 set to *CHOOSE boxes*, the ALG menu shows the following functions:


ALG MENU
1.COLLECT
2.EXPAND
3.FACTOR
4.LNCOLLECT
5.LIN
6.PARTFRAC


ALG MENU
4.LNCOLLECT
5.LIN
6.PARTFRAC
7.SOLVE
8.SUBST
9.TEXPAND

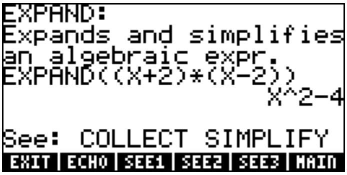
Rather than listing the description of each function in this manual, the user is invited to look up the description using the calculator's help facility: TOOL NXT HELP ENTER . To locate a particular function, type the first letter of the function. For example, for function COLLECT, we type ALPHA C , then use the up and down arrow keys, ▲▼ , to locate COLLECT within the help window.

To complete the operation press OK. Here is the help screen for function COLLECT:



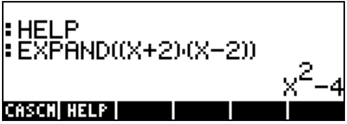
```
COLLECT:
Recursive   factoriza-
tion of a polynomial
over integers
COLLECT(X^2-4)
           (X+2)*(X-2)
See: EXPAND FACTOR
EXIT ECHO SEE1 SEE2 SEE3 MAIN
```

We notice that, at the bottom of the screen, the line See: EXPAND FACTOR suggests links to other help facility entries, the functions EXPAND and FACTOR. To move directly to those entries, press the soft menu key SEE1 for EXPAND, and SEE2 for FACTOR. Pressing SEE1, for example, shows the following information for EXPAND:



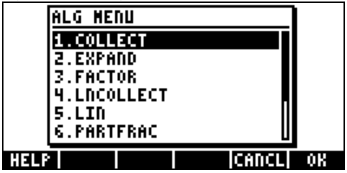
```
EXPAND:
Expands and simplifies
an algebraic expr.
EXPAND((X+2)*(X-2))
                 X^2-4

See: COLLECT SIMPLIFY
EXIT ECHO SEE1 SEE2 SEE3 MAIN
```

The help facility provides not only information on each command, but also provides an example of its application. This example can be copied onto your stack by pressing the ECHO soft menu key. For example, for the EXPAND entry shown above, press the ECHO soft menu key to get the following example copied to the stack (press ENTER to execute the command):

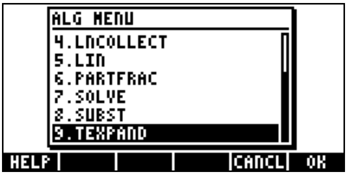


```
:HELP
:EXPAND((X+2)·(X-2))
                  X²-4
CASCM HELP
```

Thus, we leave for the user to explore the applications of the functions in the ALG (or ALGB) menu. This is a list of the commands:



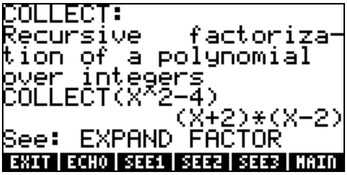
```
ALG MENU
1.COLLECT
2.EXPAND
3.FACTOR
4.LNCOLLECT
5.LIN
6.PARTFRAC
HELP        CANCL OK
```



```
ALG MENU
4.LNCOLLECT
5.LIN
6.PARTFRAC
7.SOLVE
8.SUBST
9.TEXPAND
HELP        CANCL OK
```

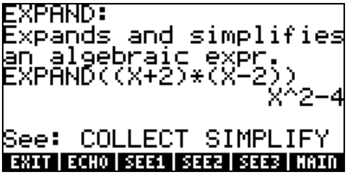
The help facility will show the following information on the commands:

COLLECT:



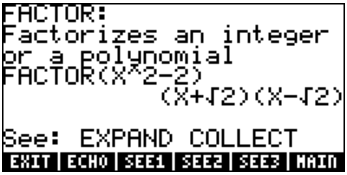
```
COLLECT:
Recursive   factoriza-
tion of a polynomial
over integers
COLLECT(X^2-4)
          (X+2)*(X-2)
See: EXPAND FACTOR
```

EXPAND:



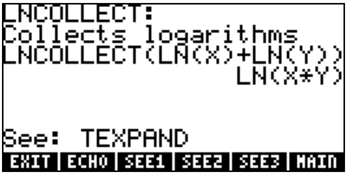
```
EXPAND:
Expands and simplifies
an algebraic expr.
EXPAND((X+2)*(X-2))
                 X^2-4

See: COLLECT SIMPLIFY
```

FACTOR:



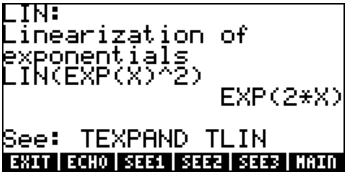
```
FACTOR:
Factorizes an integer
or a polynomial
FACTOR(X^2-2)
           (X+√2)(X-√2)

See: EXPAND COLLECT
```

LNCOLLECT:



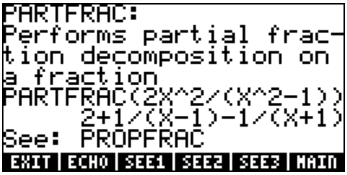
```
LNCOLLECT:
Collects logarithms
LNCOLLECT(LN(X)+LN(Y))
              LN(X*Y)


See: TEXPAND
```

LIN:



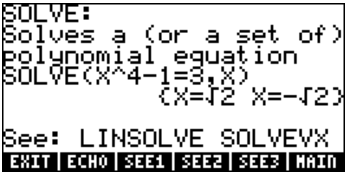
```
LIN:
Linearization of
exponentials
LIN(EXP(X)^2)
              EXP(2*X)

See: TEXPAND TLIN
```

PARTFRAC:



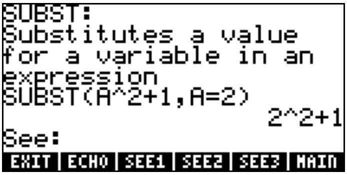
```
PARTFRAC:
Performs partial frac-
tion decomposition on
a fraction
PARTFRAC(2X^2/(X^2-1))
     2+1/(X-1)-1/(X+1)
See: PROPFRAC
```

SOLVE:



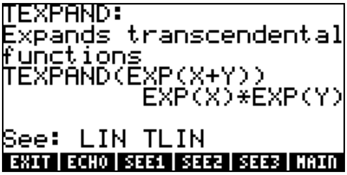
```
SOLVE:
Solves a (or a set of)
polynomial equation
SOLVE(X^4-1=3,X)
           {X=√2 X=-√2}

See: LINSOLVE SOLVEVX
```

SUBST:



```
SUBST:
Substitutes a value
for a variable in an
expression
SUBST(A^2+1,A=2)
                2^2+1
See:
```

TEXPAND:



```
TEXPAND:
Expands transcendental
functions
TEXPAND(EXP(X+Y))
         EXP(X)*EXP(Y)

See: LIN TLIN
```

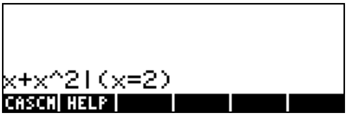
**Note:** Recall that, to use these, or any other functions in the RPN mode, you must enter the argument first, and then the function. For example, the example for TEXPAND, in RPN mode will be set up as:

' ↰ $e^x$ + ALPHA X + ALPHA Y ENTER

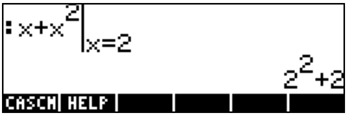
At this point, select function TEXPAND from menu ALG (or directly from the catalog ↱ CAT ), to complete the operation.

## Other forms of substitution in algebraic expressions

Functions SUBST, shown above, is used to substitute a variable in an expression. A second form of substitution can be accomplished by using the ↱ | (associated with the I key). For example, in ALG mode, the following entry will substitute the value *x = 2* in the expression *x+x*$^2$. The figure to the left shows the way to enter the expression (the substituted value, x=2, must be enclosed in parentheses) before pressing ENTER. After the ENTER key is pressed, the result is shown in the right-hand figure:



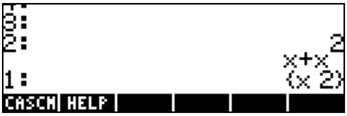
```
x+x^2|(x=2)
```



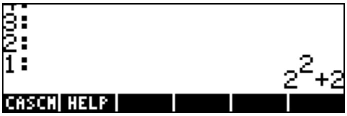
$:x+x^2|_{x=2}$

$2^2+2$

In RPN mode, this can be accomplished by entering first the expression where the substitution will be performed (x+x$^2$), followed by a list (see Chapter 8) containing the substitution variable, an space, and the value to be substituted, i.e., {x 2}. The final step is to press the keystroke combination: ↱ | .



```
3:
2:                x+x^2
1:                {x 2}
```

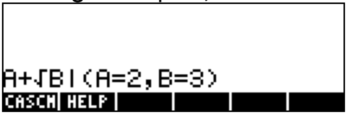


```
3:
2:
1:                2^2+2
```

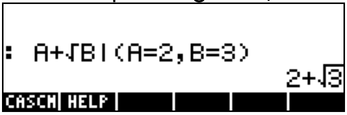
The required keystrokes are the following:

' ALPHA ↰ X + ALPHA ↰ X $Y^x$ 2 ENTER
↰ {} ALPHA ↰ X SPC 2 ENTER ↱ | ENTER

In ALG mode, substitution of more than one variable is possible as illustrated in the following example (shown before and after pressing ENTER)

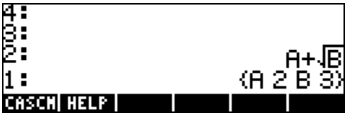


```
A+√B|(A=2,B=3)
```

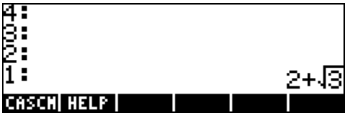


```
: A+√B|(A=2,B=3)
                    2+√3
```

In RPN mode, it is also possible to substitute more than one variable at a time, as illustrated in the example below. Recall that RPN mode uses a list of variable names and values for substitution.

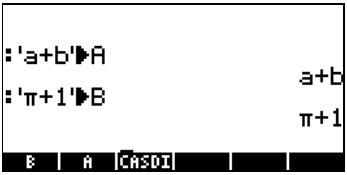


```
4:
3:
2:                  A+√B
1:             {A 2 B 3}
```



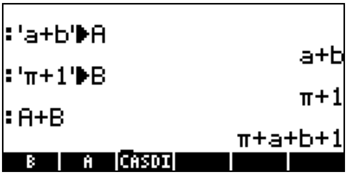
```
4:
3:
2:
1:                  2+√3
```

A different approach to substitution consists in defining the substitution expressions in calculator variables and placing the name of the variables in the original expression. For example, in ALG mode, store the following variables:



```
:'a+b'▶A
                     a+b
:'π+1'▶B
                     π+1
```

Then, enter the expression A+B:



```
:'a+b'▶A
                     a+b
:'π+1'▶B
                     π+1
:A+B
                 π+a+b+1
```

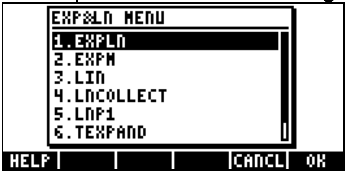
The last expression entered is automatically evaluated after pressing the ENTER key, producing the result shown above.

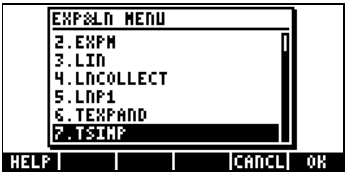
## Operations with transcendental functions

The calculator offers a number of functions that can be used to replace expressions containing logarithmic, exponential, trigonometric, and hyperbolic functions in terms of trigonometric identities or in terms of exponential functions. The menus containing functions to replace trigonometric functions can be obtained directly from the keyboard by pressing the right-shift key followed by the 8 key, i.e., ↱ TRIG . The combination of this key with the left-shift key, i.e., ↱ EXP&LN , produces a menu that lets you replace expressions in terms of exponential or natural logarithm functions. In the next sections we cover those menus in more detail.

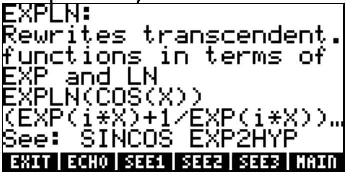
## Expansion and factoring using log-exp functions

The ↰ EXP&LN produces the following menu:

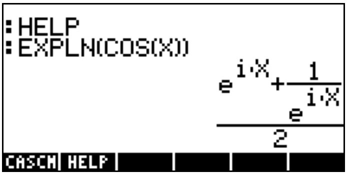




Information and examples on these commands are available in the help facility of the calculator. Some of the command listed in the EXP&LN menu, i.e., LIN, LNCOLLECT, and TEXPAND are also contained in the ALG menu presented earlier. Functions LNP1 and EXPM were introduced in menu HYPERBOLIC, under the MTH menu (See Chapter 2). The only remaining function is EXPLN. Its description is shown in the left-hand side, the example from the help facility is shown to the right:



```
EXPLN:
Rewrites transcendent.
functions in terms of
EXP and LN
EXPLN(COS(X))
(EXP(i*X)+1/EXP(i*X))…
See: SINCOS EXP2HYP
```

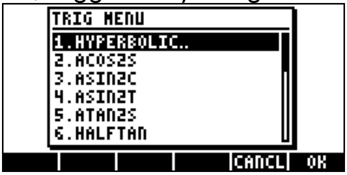


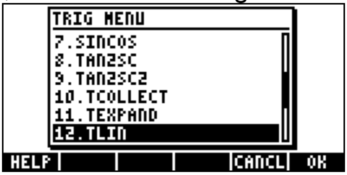
```
:HELP
:EXPLN(COS(X))
```

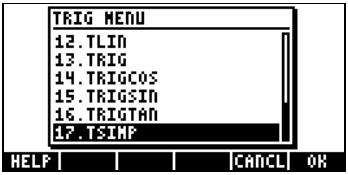
$$\frac{e^{i\cdot X}+\frac{1}{e^{i\cdot X}}}{2}$$

## Expansion and factoring using trigonometric functions

The TRIG menu, triggered by using ↱ TRIG , shows the following functions:







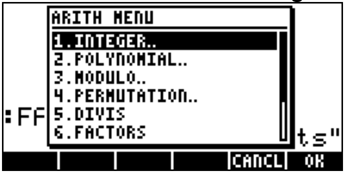
These functions allow to simplify expressions by replacing some category of trigonometric functions for another one. For example, the function ACOS2S allows to replace the function *arccosine* (acos(x)) with its expression in terms of *arcsine* (asin(x)).

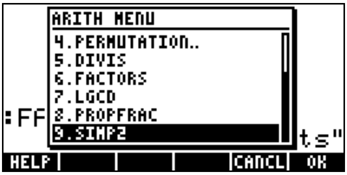
Description of these commands and examples of their applications are available in the calculator's help facility (TOOL NXT HELP). The user is invited to explore this facility to find information on the commands in the TRIG menu.

Notice that the first command in the TRIG menu is the HYPERBOLIC menu, whose functions were introduced in Chapter 2.

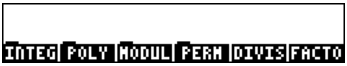
## Functions in the ARITHMETIC menu

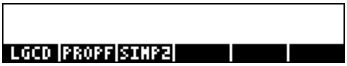
The ARITHMETIC menu contains a number of sub-menus for specific applications in number theory (integers, polynomials, etc.), as well as a number of functions that apply to general arithmetic operations. The ARITHMETIC menu is triggered through the keystroke combination ← ARITH (associated with the 1 key). With system flag 117 set to *CHOOSE boxes*, ← ARITH shows the following menu:





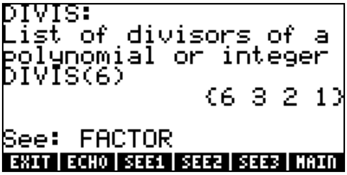
Out of this menu list, options 5 through 9 (*DIVIS, FACTORS, LGCD, PROPFRAC, SIMP2*) correspond to common functions that apply to integer numbers or to polynomials. The remaining options (*1. INTEGER*, *2. POLYNOMIAL*, *3. MODULO*, and *4. PERMUTATION*) are actually sub-menus of functions that apply to specific mathematical objects.  This distinction between sub-menus (options 1 through 4) and plain functions (options 5 through 9) is made clear when system flag 117 is set to *SOFT menus*. Activating the ARITHMETIC menu (↰ ARITH ), under these circumstances, produces:





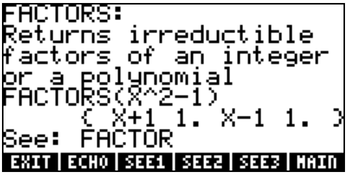
Following, we present the help facility entries for the functions of options 5 through 9 in the ARITHMETIC menu:

DIVIS:



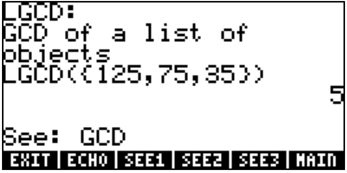
```
DIVIS:
List of divisors of a
polynomial or integer
DIVIS(6)
               {6 3 2 1}
See: FACTOR
```

FACTORS:



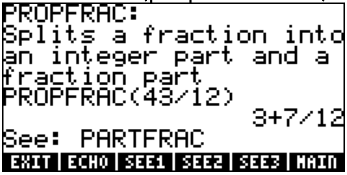
```
FACTORS:
Returns irreductible
factors of an integer
or a polynomial
FACTORS(X^2-1)
    { X+1 1. X-1 1. }
See: FACTOR
```

LGCD (Greatest Common Denominator):



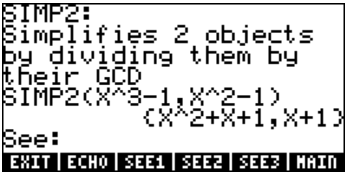
```
LGCD:
GCD of a list of
objects
LGCD({125,75,35})
                      5
See: GCD
```

PROPFRAC (proper fraction)



```
PROPFRAC:
Splits a fraction into
an integer part and a
fraction part
PROPFRAC(43/12)
                 3+7/12
See: PARTFRAC
```

SIMP2:



```
SIMP2:
Simplifies 2 objects
by dividing them by
their GCD
SIMP2(X^3-1,X^2-1)
         {X^2+X+1,X+1}
See:
```

The functions associated with the ARITHMETIC submenus: INTEGER, POLYNOMIAL, MODULO, and PERMUTATION, are the following:

## INTEGER menu

EULER Number of integers < n, co -prime with n

| | |
|---|---|
| IABCUV | Solves au + bv = c, with a,b,c = integers |
| IBERNOULLI | n-th Bernoulli number |
| ICHINREM | Chinese reminder for integers |
| IDIV2 | Euclidean division of two integers |
| IEGCD | Returns u,v, such that au + bv = gcd(a,b) |
| IQUOT | Euclidean quotient of two integers |
| IREMAINDER | Euclidean remainder of two integers |
| ISPRIME? | Test if an integer number is prime |
| NEXTPRIME | Next prime for a given integer number |
| PA2B2 | Prime number as square norm of a complex number |
| PREVPRIME | Previous prime for a given integer number |

## POLYNOMIAL menu

| | |
|---|---|
| ABCUV | Bézout polynomial equation (au+bv=c) |
| CHINREM | Chinese remainder for polynomials |
| CYCLOTOMIC | n-th cyclotomic polynomial |
| DIV2 | Euclidean division of two polynomials |
| EGDC | Returns u,v, from au+bv=gcd(a,b) |
| FACTOR | Factorizes an integer number or a polynomial |
| FCOEF | Generates fraction given roots and multiplicity |
| FROOTS | Returns roots and multiplicity given a fraction |
| GCD | Greatest common divisor of 2 numbers or polynomials |
| HERMITE | n-th degree Hermite polynomial |
| HORNER | Horner evaluation of a polynomial |
| LAGRANGE | Lagrange polynomial interpolation |
| LCM | Lowest common multiple of 2 numbers or polynomials |
| LEGENDRE | n-th degree Legendre polynomial |
| PARTFRAC | Partial-fraction decomposition of a given fraction |
| PCOEF | (help-facility entry missing) |
| PTAYL | Returns Q(x-a) in Q(x-a) = P(x), Taylor polynomial |
| QUOT | Euclidean quotient of two polynomials |
| RESULTANT | Determinant of the Sylvester matrix of 2 polynomials |
| REMAINDER | Euclidean reminder of two polynomials |
| STURM | Sturm sequences for a polynomial |
| STURMAB | Sign at low bound and number of zeros between bounds |

## MODULO menu

| | |
|---|---|
| ADDTMOD | Add two expressions modulo current modulus |
| DIVMOD | Divides 2 polynomials modulo current modulus |
| DIV2MOD | Euclidean division of 2 polynomials with modular coefficients |
| EXPANDMOD | Expands/simplify polynomial modulo current modulus |
| FACTORMOD | Factorize a polynomial modulo current modulus |
| GCDMOD | GCD of 2 polynomials modulo current modulus |
| INVMOD | inverse of integer modulo current modulus |
| MOD | (not entry available in the help facility) |
| MODSTO | Changes modulo setting to specified value |
| MULTMOD | Multiplication of two polynomials modulo current modulus |
| POWMOD | Raises polynomial to a power modulo current modulus |
| SUBTMOD | Subtraction of 2 polynomials modulo current modulus |

# Applications of the ARITHMETIC menu

This section is intended to present some of the background necessary for application of the ARITHMETIC menu functions. Definitions are presented next regarding the subjects of polynomials, polynomial fractions and modular arithmetic. The examples presented below are presented independently of the calculator setting (ALG or RPN)

## Modular arithmetic

Consider a counting system of integer numbers that periodically cycles back on itself and starts again, such as the hours in a clock. Such counting system is called a *ring*. Because the number of integers used in a ring is finite, the arithmetic in this ring is called *finite arithmetic*. Let our system of finite integer numbers consists of the numbers *0, 1, 2, 3, …, n-1, n*. We can also refer to the arithmetic of this counting system as *modular arithmetic of modulus n*. In the case of the hours of a clock, the modulus is 12. (If working with modular arithmetic using the hours in a clock, however, we would have to use the integer numbers 0, 1, 2, 3, …, 10, 11, rather than 1, 2, 3,…,11, 12).

**Operations in modular arithmetic**

*Addition in modular arithmetic* of modulus *n,* which is a positive integer, follow the rules that if *j* and *k* are any two nonnegative integer numbers, both smaller than *n*, if $j+k\geq n$, then $j+k$ is defined as $j+k-n$. For example, in the case of the clock, i.e., for *n* = 12, 6+9 "=" 3. To distinguish this 'equality' from infinite arithmetic equalities, the symbol ≡ is used in place of the equal sign, and the relationship between the numbers is referred to as a *congruence* rather than an equality. Thus, for the previous example we would write *6+9 ≡ 3 (mod 12),* and read this expression as "*six plus nine is congruent to three, modulus twelve.*" If the numbers represent the hours since midnight, for example, the congruence 6+9 ≡ 3 (mod 12), can be interpreted as saying that "six hours past the ninth hour after midnight will be three hours past noon." Other sums that can be defined in modulus 12 arithmetic are: 2+5 ≡ 7 (mod 12); 2+10 ≡ 0 (mod 12); 7+5 ≡ 0 (mod 12); etcetera.

The rule for *subtraction* will be such that if $j - k < 0$, then $j$-$k$ is defined as $j$-$k+n$. Therefore, *8-10 ≡ 2 (mod 12),* is read "*eight minus ten is congruent to two, modulus twelve.*" Other examples of subtraction in modulus 12 arithmetic would be 10-5 ≡ 5 (mod 12); 6-9 ≡ 9 (mod 12); 5 – 8 ≡ 9 (mod 12); 5 –10 ≡ 7 (mod 12); etcetera.

*Multiplication* follows the rule that if $j\cdot k > n$, so that $j\cdot k = m\cdot n + r$, where *m* and *r* are nonnegative integers, both less than *n*, then $j\cdot k \equiv r$ (mod *n*). The result of multiplying *j* times *k* in modulus *n* arithmetic is, in essence, the integer remainder of $j\cdot k/n$ in infinite arithmetic, if $j\cdot k>n$. For example, in modulus 12 arithmetic we have 7·3 = 21 = 12 + 9, (or, 7·3/12 = 21/12 = 1 + 9/12, i.e., the integer reminder of 21/12 is 9). We can now write 7·3 ≡ 9 (mod 12), and read the latter result as "*seven times three is congruent to nine, modulus twelve.*"

The operation of *division* can be defined in terms of multiplication as follows, $r/k \equiv j$ (mod *n*), if, $j\cdot k \equiv r$ (mod *n*). This means that *r* must be the remainder of $j\cdot k/n$. For example, 9/7 ≡ 3 (mod 12), because 7·3 ≡ 9 (mod 12). Some divisions are not permitted in modular arithmetic. For example, in modulus 12 arithmetic you cannot define 5/6 (mod 12) because the multiplication table of

6 does not show the result 5 in modulus 12 arithmetic. This multiplication table is shown below:

| | | | |
|---|---|---|---|
| 6*0 (mod 12) | 0 | 6*6 (mod 12) | 0 |
| 6*1 (mod 12) | 6 | 6*7 (mod 12) | 6 |
| 6*2 (mod 12) | 0 | 6*8 (mod 12) | 0 |
| 6*3 (mod 12) | 6 | 6*9 (mod 12) | 6 |
| 6*4 (mod 12) | 0 | 6*10 (mod 12) | 0 |
| 6*5 (mod 12) | 6 | 6*11 (mod 12) | 6 |

**Formal definition of a finite arithmetic ring**

The expression $a \equiv b \ (mod\ n)$ is interpreted as "*a* is congruent to *b*, modulo *n*," and holds if *(b-a)* is a multiple of *n*. With this definition the rules of arithmetic simplify to the following:

If $a \equiv b \ (mod\ n)$ and $c \equiv d \ (mod\ n)$,

then

$$a+c \equiv b+d \ (mod\ n),$$
$$a\text{-}c \equiv b - d \ (mod\ n),$$
$$a\times c \equiv b\times d \ (mod\ n).$$

For division, follow the rules presented earlier. For example, 17 ≡ 5 (mod 6), and 21 ≡ 3 (mod 6). Using these rules, we can write:

17 + 21 ≡ 5 + 3 (mod 6) => 38 ≡ 8 (mod 6) => 38 ≡ 2 (mod 6)
17 – 21 ≡ 5 - 3 (mod 6) => -4 ≡ 2 (mod 6)
17 × 21 ≡ 5 × 3 (mod 6) => 357 ≡ 15 (mod 6) => 357 ≡ 3 (mod 6)

Notice that, whenever a result in the right-hand side of the "congruence" symbol produces a result that is larger than the modulo (in this case, *n* = 6), you can always subtract a multiple of the modulo from that result and simplify it to a number smaller than the modulo. Thus, the results in the first case *8 (mod 6)* simplifies to *2 (mod 6*), and the result of the third case, *15 (mod 6)* simplifies to *3 (mod 6)*. Confusing? Well, not if you let the calculator handle those operations. Thus, read the following section to understand how finite arithmetic rings are operated upon in your calculator.

## Finite arithmetic rings in the calculator

All along we have defined our finite arithmetic operation so that the results are always positive. The modular arithmetic system in the calculator is set so that the ring of modulus n includes the numbers *-n/2+1, …,-1, 0, 1,…,n/2-1, n/2*, if n is even, and *–(n-1)/2, -(n-3)/2,…,-1,0,1,…,(n-3)/2, (n-1)/2*, if n is odd. For example, for n = 8 (even), the finite arithmetic ring in the calculator includes the numbers: (-3,-2,-1,0,1,3,4), while for n = 7 (odd), the corresponding calculator's finite arithmetic ring is given by (-3,-2,-1,0,1,2,3).

### Modular arithmetic in the calculator

To launch the modular arithmetic menu in the calculator select the MODULO sub-menu within the ARITHMETIC menu (↰ *ARITH* ). The available menu includes functions: ADDTMOD, DIVMOD, DIV2MOD, EXPANDMOD, FACTORMOD, GCDMOD, INVMOD, MOD, MODSTO, MULTMOD, POWMOD, and SUBTMOD. Brief descriptions of these functions were provided in an earlier section. Next we present some applications of these functions.

### Setting the modulus (or MODULO)

The calculator contains a variable called MODULO that is placed in the {HOME CASDIR} directory and will store the magnitude of the modulus to be used in modular arithmetic.

The default value of MODULO is 13. To change the value of MODULO, you can either store the new value directly in the variable MODULO in the sub-directory {HOME CASDIR} Alternatively, you can store a new MODULO value by using function MODSTO.

### Modular arithmetic operations with numbers

To add, subtract, multiply, divide, and raise to a power using modular arithmetic you will use the functions ADDTMOD, SUBTMOD, MULTMOD, DIV2MOD and DIVMOD (for division), and POWMOD. In RPN mode you need to enter the two numbers to operate upon, separated by an [ENTER] or an [SPC] entry, and then press the corresponding modular arithmetic function. For example, using a modulus of 12, try the following operations:

### *ADDTMOD examples*

$6+5 \equiv -1 \pmod{12}$ $6+6 \equiv 0 \pmod{12}$ $6+7 \equiv 1 \pmod{12}$
$11+5 \equiv 4 \pmod{12}$ $8+10 \equiv -6 \pmod{12}$

### *SUBTMOD examples*

$5 - 7 \equiv -2 \pmod{12}$ $8 - 4 \equiv 4 \pmod{12}$ $5 - 10 \equiv -5 \pmod{12}$
$11 - 8 \equiv 3 \pmod{12}$ $8 - 12 \equiv -4 \pmod{12}$

### *MULTMOD examples*

$6 \cdot 8 \equiv 0 \pmod{12}$ $9 \cdot 8 \equiv 0 \pmod{12}$ $3 \cdot 2 \equiv 6 \pmod{12}$
$5 \cdot 6 \equiv 6 \pmod{12}$ $11 \cdot 3 \equiv -3 \pmod{12}$

### *DIVMOD examples*

$12/3 \equiv 4 \pmod{12}$ $12/8 \pmod{12}$ does not exist
$25/5 \equiv 5 \pmod{12}$ $64/13 \equiv 4 \pmod{12}$
$66/6 \equiv -1 \pmod{12}$

### *DIV2MOD examples*

$2/3 \pmod{12}$ does not exist
$26/12 \pmod{12}$ does not exist
$125/17 \pmod{12} \equiv 1$ with remainder = 0
$68/7 \equiv -4 \pmod{12}$ with remainder = 0
$7/5 \equiv -1 \pmod{12}$ with remainder = 0

**Note**: DIVMOD provides the quotient of the modular division j/k (mod n), while DIMV2MOD provides no only the quotient but also the remainder of the modular division j/k (mod n).

### *POWMOD examples*

$2^3 \equiv -4 \pmod{12}$ $3^5 \equiv 3 \pmod{12}$ $5^{10} \equiv 1 \pmod{12}$
$11^8 \equiv 1 \pmod{12}$ $6^2 \equiv 0 \pmod{12}$ $9^9 \equiv -3 \pmod{12}$

In the examples of modular arithmetic operations shown above, we have used numbers that not necessarily belong to the ring, i.e., numbers such as 66, 125, 17, etc. The calculator will convert those numbers to ring numbers before operating on them. You can also convert any number into a ring number by using the function EXPANDMOD. For example,

EXPANDMOD(125) ≡ 5 (mod 12)
EXPANDMOD(17) ≡ 5 (mod 12)
EXPANDMOD(6) ≡ 6 (mod 12)

### *The modular inverse of a number*

Let a number *k* belong to a finite arithmetic ring of modulus *n*, then the modular inverse of *k*, i.e., *1/k* (mod *n)*, is a number *j*, such that *j·k ≡ 1 (*mod *n*). The modular inverse of a number can be obtained by using the function INVMOD in the MODULO sub-menu of the ARITHMETIC menu. For example, in modulus 12 arithmetic:

1/6 (mod 12) does not exist.        1/5 ≡ 5 (mod 12)
1/7 ≡ -5 (mod 12)        1/3 (mod 12) does not exist.
1/11 ≡ -1 (mod 12)

### *The MOD operator*

The MOD operator is used to obtain the ring number of a given modulus corresponding to a given integer number. On paper this operation is written as *m* mod *n = p*, and is read as "*m* modulo *n* is equal to *p*". For example, to calculate 15 mod 8, enter:

- ALG mode: 1 5 MOD 8 ENTER
- RPN mode: 1 5 ENTER 8 ENTER MOD

The result is 7, i.e., 15 mod 8 = 7. Try the following exercises:
18 mod 11 = 7        23 mod 2 =1        40 mod 13 = 1
23 mod 17 = 6        34 mod 6 = 4

One practical application of the MOD function for programming purposes is to determine when an integer number is odd or even, since *n* mod 2 = 0, if *n* is even, and *n* mode 2 = 1, if *n* is odd. It can also be used to determine when an integer *m* is a multiple of another integer *n*, for if that is the case *m* mod *n* = 0.

**Note:** Refer to the help facility in the calculator for description and examples on other modular arithmetic. Many of these functions are applicable to polynomials. For information on modular arithmetic with polynomials please refer to a textbook on number theory.

## Polynomials

Polynomials are algebraic expressions consisting of one or more terms containing decreasing powers of a given variable. For example, 'X^3+2*X^2-3*X+2' is a third-order polynomial in X, while 'SIN(X)^2-2' is a second-order polynomial in SIN(X). A listing of polynomial-related functions in the ARITHMETIC menu was presented earlier. Some general definitions on polynomials are provided next. In these definitions *A(X), B(X), C(X), P(X), Q(X), U(X), V(X),* etc., are polynomials.

- Polynomial fraction: a fraction whose numerator and denominator are polynomials, say, C(X) = A(X)/B(X)
- Roots, or zeros, of a polynomial: values of X for which P(X) = 0
- Poles of a fraction: roots of the denominator
- Multiplicity of roots or poles: the number of times a root shows up, e.g., $P(X) = (X+1)^2(X-3)$ has roots {-1, 3} with multiplicities {2,1}
- Cyclotomic polynomial ($P_n(X)$): a polynomial of order EULER(*n)* whose roots are the primitive *n*-th roots of unity, e.g., $P_2(X) = X+1$, $P_4(X) = X^2+1$
- Bézout's polynomial equation: A(X) U(X) + B(X)V(X) = C(X)

Specific examples of polynomial applications are provided next.

### Modular arithmetic with polynomials

The same way that we defined a finite-arithmetic ring for numbers in a previous section, we can define a finite-arithmetic ring for polynomials with a given polynomial as modulo. For example, we can write a certain polynomial *P(X)* as *P(X) = X (mod* $X^2$*),* or another polynomial *Q(X) = X + 1 (mod X-2).*

A polynomial, *P(X)* belongs to a finite arithmetic ring of polynomial modulus *M(X),* if there exists a third polynomial *Q(X),* such that *(P(X) – Q(X))* is a multiple of *M(X).* We then would write: *P(X) ≡ Q(X) (mod M(X)).* The later expression is interpreted as *"P(X) is congruent to Q(X), modulo M(X)".*

### The CHINREM function

CHINREM stands for CHINese REMainder. The operation coded in this command solves a system of two congruences using the Chinese Remainder Theorem. This command can be used with polynomials, as well as with integer numbers (function ICHINREM). The input consists of two vectors *[expression_1, modulo_1] and [expression_2, modulo_2]*. The output is a vector *containing [expression_3, modulo_3]*, where *modulo_3* is related to the product *(modulo_1)·(modulo_2)*. Example: CHINREM(['X+1', 'X^2-1'],['X+1','X^2']) = ['X+1',-(X^4-X^2)]

#### *Statement of the Chinese Remainder Theorem for integers*

If $m_1$, $m_2$,…,$m_r$ are natural numbers every pair of which are relatively prime, and $a_1$, $a_2$, …, $a_r$ are any integers, then there is an integer *x* that simultaneously satisfies the congruences: $x \equiv a_1$ (mod $m_1$), $x \equiv a_2$ (mod $m_2$), …, $x \equiv a_r$ (mod $m_r$). Additionally, if *x* = *a* is any solution then all other solutions are congruent to a modulo equal to the product $m_1 \cdot m_2 \cdot \ldots m_r$.

### The EGCD function

EGCD stands for Extended Greatest Common Divisor. Given two polynomials, A(X) and B(X), function EGCD produces the polynomials C(X), U(X), and V(X), so that C(X) = U(X)*A(X) + V(X)*B(X). For example, for A(X) = X^2+1, B(X) = X^2-1, EGCD(A(X),B(X)) = {2, 1, -1}. i.e., 2 = 1*( X^2+1')-1*( X^2-1). Also, EGCD('X^3-2*X+5','X') = { 5, '-(X^2-2)', 1}, i.e., 5 = – (X^2-2)*X + 1*(X^3-2*X+5).

### The GCD function

The function GCD (Greatest Common Denominator) can be used to obtain the greatest common denominator of two polynomials or of two lists of polynomials of the same length. The two polynomials or lists of polynomials will be placed in stack levels 2 and 1 before using GCD. The results will be a polynomial or a list representing the greatest common denominator of the two polynomials or of each list of polynomials. Examples, in RPN mode, follow (calculator set to Exact mode):

'X^3-1'ENTER'X^2-1'ENTER GCD Results in: 'X-1'

{'X^2+2*X+1','X^3+X^2'} ENTER {'X^3+1','X^2+1'} ENTER GCD results in {'X+1' 1}

### The HERMITE function

The function HERMITE [HERMI] uses as argument an integer number, k, and returns the Hermite polynomial of k-th degree. A Hermite polynomial, $He_k(x)$ is defined as

$$He_0 = 1, \quad He_n(x) = (-1)^n e^{x^2/2} \frac{d^n}{dx^n}(e^{-x^2/2}), \quad n = 1,2,...$$

An alternate definition of the Hermite polynomials is

$$H_0* = 1, \quad H_n*(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}(e^{-x^2}), \quad n = 1,2,...$$

where $d^n/dx^n$ = n-th derivative with respect to x. This is the definition used in the calculator.

Examples: The Hermite polynomials of orders 3 and 5 are given by:

HERMITE(3) = '8*X^3-12*X' ,

And HERMITE(5) = '32*x^5-160*X^3+120*X'.

### The HORNER function

The function HORNER produces the Horner division, or synthetic division, of a polynomial P(X) by the factor (X-*a*). The input to the function are the polynomial P(X) and the number *a*. The function returns the quotient polynomial Q(X) that results from dividing P(X) by (X-*a*), the value of *a*, and the value of P(*a*), in that order. In other words, P(X) = Q(X)(X-a)+P(a). For example, HORNER('X^3+2*X^2-3*X+1',2) = {'X^2+4*X+5', 2, 11}. We could, therefore, write $X^3+2X^2-3X+1 = (X^2+4X+5)(X-2)+11$. A second example: HORNER('X^6-1',-5)= {'X^5-5*X^4+25*X^3-125*X^2+625*X-3125',-5, 15624} i.e., $X^6-1 = (X^5-5*X^4+25X^3-125X^2+625X-3125)(X+5)+15624$.

### The variable VX

A variable called VX exists in the calculator's {HOME CASDIR} directory that takes, by default, the value of 'X'. This is the name of the preferred independent variable for algebraic and calculus applications. Avoid using the variable VX in your programs or equations, so as to not get it confused with the CAS' VX. If you need to refer to the x-component of velocity, for example, you can use vx or Vx. For additional information on the CAS variable see Appendix C.

### The LAGRANGE function

The function LAGRANGE requires as input a matrix having two rows and *n* columns. The matrix stores data points of the form [[$x_1$,$x_2$, …, $x_n$] [$y_1$, $y_2$, …, $y_n$]]. Application of the function LAGRANGE produces the polynomial expanded from

$$p_{n-1}(x) = \sum_{j=1}^{n} \frac{\prod_{k=1,k\neq j}^{n}(x - x_k)}{\prod_{k=1,k\neq j}^{n}(x_j - x_k)} \cdot y_j.$$

For example, for n = 2, we will write:

$$p_1(x) = \frac{x - x_2}{x_1 - x_2} \cdot y_1 + \frac{x - x_1}{x_2 - x_1} \cdot y_2 = \frac{(y_1 - y_2) \cdot x + (y_2 \cdot x_1 - y_1 \cdot x_2)}{x_1 - x_2}$$

Check this result with your calculator:
LAGRANGE([[ x1,x2],[y1,y2]]) = '((y1-y2)*X+(y2*x1-y1*x2))/(x1-x2)'.

Other examples: LAGRANGE([[1, 2, 3][2, 8, 15]]) = '(X^2+9*X-6)/2'
LAGRANGE([[0.5,1.5,2.5,3.5,4.5][12.2,13.5,19.2,27.3,32.5]]) =
'-(.1375*X^4+ -.7666666666667*X^3+ - .74375*X^2 =
1.991666666667*X-12.92265625)'.

**Note:** Matrices are introduced in Chapter 10.

### The LCM function

The function LCM (Least Common Multiple) obtains the least common multiple of two polynomials or of lists of polynomials of the same length. Examples:

LCM('2*X^2+4*X+2' ,'X^2-1' ) = '(2*X^2+4*X+2)*(X-1)'.
LCM('X^3-1','X^2+2*X') = '(X^3-1)*( X^2+2*X)'

### The LEGENDRE function

A Legendre polynomial of order n is a polynomial function that solves the differential equation $(1-x^2)\cdot\frac{d^2y}{dx^2}-2\cdot x\cdot\frac{dy}{dx}+n\cdot(n+1)\cdot y=0$

To obtain the *n*-th order Legendre polynomial, use LEGENDRE(*n*), e.g.,

LEGENDRE(3) = '(5*X^3-3*X)/2'
LEGENDRE(5) = '(63*X ^5-70*X^3+15*X)/8'

### The PCOEF function

Given an array containing the roots of a polynomial, the function PCOEF generates an array containing the coefficients of the corresponding polynomial. The coefficients correspond to decreasing order of the independent variable. For example: PCOEF([-2,–1,0,1,1,2]) = [1. –1. –5. 5. 4. –4. 0.], which represents the polynomial $X^6$-$X^5$-5$X^4$+5$X^3$+4$X^2$-4X.

### The PROOT function

Given an array containing the coefficients of a polynomial, in decreasing order, the function PROOT provides the roots of the polynomial. Example, from $X^2$+5X-6 =0, PROOT([1 –5 6]) = [2. 3.].

### The PTAYL function

Given a polynomial P(X) and a number *a*, the function PTAYL is used to obtain an expression Q(X-*a*) = P(X), i.e., to develop a polynomial in powers of (X- *a*). This is also known as a Taylor polynomial, from which the name of the function, Polynomial & TAYLor, follow.

For example, PTAYL('X^3-2*X+2',2) = 'X^3+6*X^2+10*X+6'.

In actuality, you should interpret this result to mean

'(X-2) ^3+6*(X-2) ^2+10*(X-2) +6'.

Let's check by using the substitution: 'X = x – 2'. We recover the original polynomial, but in terms of lower-case x rather than upper-case x.

### The QUOT and REMAINDER functions

The functions QUOT and REMAINDER provide, respectively, the quotient Q(X) and the remainder R(X), resulting from dividing two polynomials, $P_1$(X) and $P_2$(X). In other words, they provide the values of Q(X) and R(X) from $P_1(X)/P_2(X) = Q(X) + R(X)/P_2(X)$. For example,

QUOT(X^3-2*X+2, X-1) = X^2+X-1
REMAINDER(X^3-2*X+2, X-1) = 1.

Thus, we can write: $(X^3-2X+2)/(X-1) = X^2+X-1 + 1/(X-1)$.

> **Note**: you could get the latter result by using PROPFRAC:
> PROPFRAC('(X^3-2*X+2)/(X-1)') = 'X^2+X-1 + 1/(X-1)'.

### The EPSX0 function and the CAS variable EPS

The variable $\varepsilon$ (epsilon) is typically used in mathematical textbooks to represent a very small number. The calculator's CAS creates a variable EPS, with default value 0.0000000001 = $10^{-10}$, when you use the EPSX0 function. You can change this value, once created, if you prefer a different value for EPS. The function EPSX0, when applied to a polynomial, will replace all coefficients whose absolute value is less than EPS with a zero. Function EPSX0 is not available in the ARITHMETIC menu, it must be accessed from the function catalog (N). Example:

EPSX0('X^3-1.2E-12*X^2+1.2E-6*X+6.2E-11)=
'X^3-0*X^2+.0000012*X+0'.

With EVAL: 'X^3+.0000012*X+0'.

### The PEVAL function

The functions PEVAL (Polynomial EVALuation) can be used to evaluate a polynomial $p(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0$, given an array of coefficients $[a_n, a_{n-1}, \ldots\ a_2, a_1, a_0]$ and a value of $x_0$. The result is the evaluation $p(x_0)$. Function PEVAL is not available in the ARITHMETIC menu, it must be accessed from the function catalog (‚N). Example:

PEVAL([1,5,6,1],5) = 281.

### The TCHEBYCHEFF function

The function TCHEBYCHEFF(*n*) generates the Tchebycheff (or Chebyshev) polynomial of the first kind, order n, defined as $T_n(X) = cos(n \cdot arccos(X))$. If the integer n is negative (n < 0), the function TCHEBYCHEFF(*n*) generates the Tchebycheff polynomial of the second kind, order n, defined as $T_n(X) = sin(n \cdot arccos(X))/sin(arccos(X))$. Examples:

TCHEBYCHEFF(3) = 4*X^3-3*X
TCHEBYCHEFF(-3) = 4*X^2-1

## Fractions

Fractions can be expanded and factored by using functions EXPAND and FACTOR, from the ALG menu (,×). For example:

EXPAND('(1+X)^3/((X-1)(X+3))') = '(X^3+3*X^2+3*X+1)/(X^2+2*X-3)'
EXPAND('(X^2*(X+Y)/(2*X-X^2)^2') = '(X+Y)/(X^2-4*X+4)'
EXPAND('X*(X+Y)/(X^2-1)') = '(X^2+Y*X)/(X^2-1)'
EXPAND('4+2*(X-1)+3/((X-2)*(X+3))-5/X^2') =
'(2*X^5+4*X^4-10*X^3-14*X^2-5*X)/(X^4+X^3-6*X^2)'

FACTOR('(3*X^3-2*X^2)/(X^2-5*X+6)') = 'X^2*(3*X-2)/((X-2)*(X-3))'
FACTOR('(X^3-9*X)/(X^2-5*X+6)' ) = 'X*(X+3)/(X-2)'
FACTOR('(X^2-1)/(X^3*Y-Y)') = '(X+1)/((X^2+X+1)*Y)'

### The SIMP2 function

Functions SIMP2 and PROPFRAC are used to simplify a fraction and to produce a proper fraction, respectively. Function SIMP2 takes as arguments two numbers or polynomials, representing the numerator and denominator of a rational fraction, and returns the simplified numerator and denominator. For example: SIMP2('X^3-1','X^2-4*X+3') = { 'X^2+X+1','X-3'}.

### The PROPFRAC function

The function PROPFRAC converts a rational fraction into a "proper" fraction, i.e., an integer part added to a fractional part, if such decomposition is possible. For example:

PROPFRAC(‘5/4’) = ‘1+1/4’
PROPFRAC(‘(x^2+1)/x^2’) = ‘1+1/x^2’

### The PARTFRAC function

The function PARTFRAC decomposes a rational fraction into the partial fractions that produce the original fraction. For example:

PARTFRAC(‘(2*X^6-14*X^5+29*X^4-37*X^3+41*X^2-16*X+5)/(X^5-7*X^4+11*X^3-7*X^2+10*X)’) =
‘2*X+(1/2/(X-2)+5/(X-5)+1/2/X+X/(X^2+1))’

This technique is useful in calculating integrals (see chapter on calculus) of rational fractions.

If you have the Complex mode active, the result will be:

‘2*X+(1/2/(X+i)+1/2/(X-2)+5/(X-5)+1/2/X+1/2/(X-i))’

### The FCOEF function

The function FCOEF is used to obtain a rational fraction, given the roots and poles of the fraction.

> **Note**: If a rational fraction is given as F(X) = N(X)/D(X), the roots of the fraction result from solving the equation N(X) = 0, while the poles result from solving the equation D(X) = 0.

The input for the function is a vector listing the roots followed by their multiplicity (i.e., how many times a given root is repeated), and the poles followed by their multiplicity represented as a negative number. For example, if we want to create a fraction having roots 2 with multiplicity 1, 0 with multiplicity 3, and -5 with multiplicity 2, and poles 1 with multiplicity 2 and –3 with multiplicity 5, use:

FCOEF([2 1 0 3 –5 2 1 -2 -3 -5]) = ‘(X--5)^2*X^3*(X-2)/(X--3)^5*(X-1)^2’

If you press EVAL you will get:

‘(X^6+8*X^5+5*X^4-50*X^3)/(X^7+13*X^6+61*X^5+105*X^4-45*X^3-297*X^2-81*X+243)’

**The FROOTS function**

The function FROOTS obtains the roots and poles of a fraction. As an example, applying function FROOTS to the result produced above, will result in: [1 –2 –3 –5 0 3 2 1 –5 2]. The result shows poles followed by their multiplicity as a negative number, and roots followed by their multiplicity as a positive number. In this case, the poles are (1, -3) with multiplicities (2,5) respectively, and the roots are (0, 2, -5) with multiplicities (3, 1, 2), respectively.

Another example is: FROOT(‘(X^2-5*X+6)/(X^5-X^2)’) = [0,–2,1, –1,3,1,2, 1], i.e., poles = 0 (2), 1(1), and roots = 3(1), 2(1). If you have had Complex mode selected, then the results would be: [0 –2 1 –1 ‘-((1+i*√3)/2’ –1].

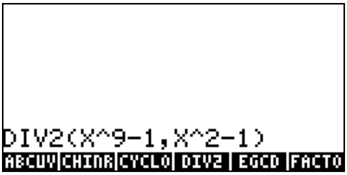
## Step-by-step operations with polynomials and fractions

By setting the CAS modes to Step/step the calculator will show simplifications of fractions or operations with polynomials in a step-by-step fashion. This is very useful to see the steps of a synthetic division. The example of dividing

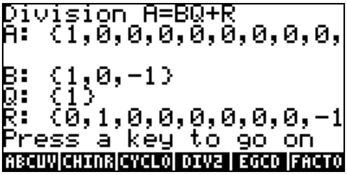
$$\frac{X^3-5X^2+3X-2}{X-2}$$

is shown in detail in Appendix C. The following example shows a lengthier synthetic division:

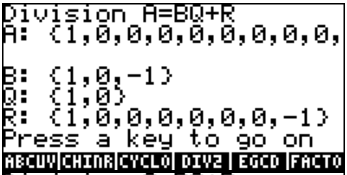
$$\frac{X^9-1}{X^2-1}$$



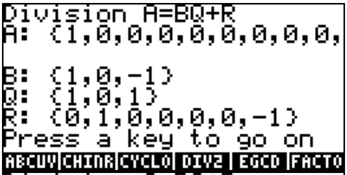
```
DIV2(X^9-1,X^2-1)
```



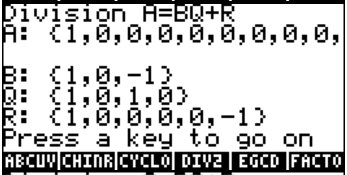
```
Division A=BQ+R
A: {1,0,0,0,0,0,0,0,0,
B: {1,0,-1}
Q: {1}
R: {0,1,0,0,0,0,0,0,-1
Press a key to go on
```



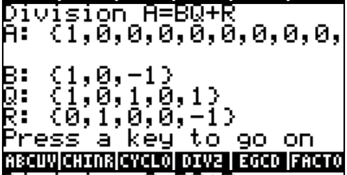
```
Division A=BQ+R
A: {1,0,0,0,0,0,0,0,0,

B: {1,0,-1}
Q: {1,0}
R: {1,0,0,0,0,0,0,-1}
Press a key to go on
ABCUV|CHINR|CYCLO| DIV2 | EGCD |FACTO
```



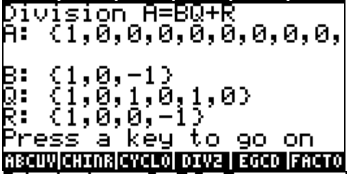
```
Division A=BQ+R
A: {1,0,0,0,0,0,0,0,0,

B: {1,0,-1}
Q: {1,0,1}
R: {0,1,0,0,0,0,-1}
Press a key to go on
ABCUV|CHINR|CYCLO| DIV2 | EGCD |FACTO
```



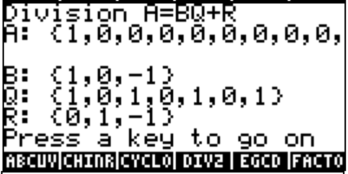
```
Division A=BQ+R
A: {1,0,0,0,0,0,0,0,0,

B: {1,0,-1}
Q: {1,0,1,0}
R: {1,0,0,0,0,-1}
Press a key to go on
ABCUV|CHINR|CYCLO| DIV2 | EGCD |FACTO
```



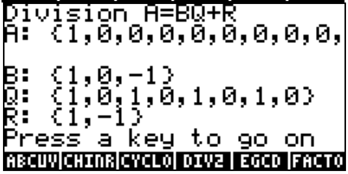
```
Division A=BQ+R
A: {1,0,0,0,0,0,0,0,0,

B: {1,0,-1}
Q: {1,0,1,0,1}
R: {0,1,0,0,-1}
Press a key to go on
ABCUV|CHINR|CYCLO| DIV2 | EGCD |FACTO
```



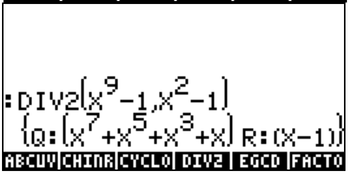
```
Division A=BQ+R
A: {1,0,0,0,0,0,0,0,0,

B: {1,0,-1}
Q: {1,0,1,0,1,0}
R: {1,0,0,-1}
Press a key to go on
ABCUV|CHINR|CYCLO| DIV2 | EGCD |FACTO
```



```
Division A=BQ+R
A: {1,0,0,0,0,0,0,0,0,

B: {1,0,-1}
Q: {1,0,1,0,1,0,1}
R: {0,1,-1}
Press a key to go on
ABCUV|CHINR|CYCLO| DIV2 | EGCD |FACTO
```



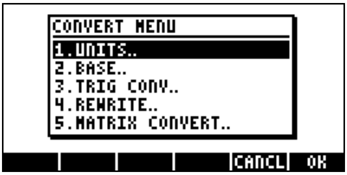
```
Division A=BQ+R
A: {1,0,0,0,0,0,0,0,0,

B: {1,0,-1}
Q: {1,0,1,0,1,0,1,0}
R: {1,-1}
Press a key to go on
ABCUV|CHINR|CYCLO| DIV2 | EGCD |FACTO
```



```
:DIV2(X^9-1,X^2-1)
 {Q:(X^7+X^5+X^3+X) R:(X-1)}
ABCUV|CHINR|CYCLO| DIV2 | EGCD |FACTO
```

## The CONVERT Menu and algebraic operations

The CONVERT menu is activated by using ↰ CONVERT key (the 6 key). This menu summarizes all conversion menus in the calculator. The list of these menus is shown next:



```
CONVERT MENU
1.UNITS..
2.BASE..
3.TRIG CONV..
4.REWRITE..
5.MATRIX CONVERT..
    |    |    |    |CANCL| OK
```

The functions available in each of the sub-menus are shown next.

### UNITS convert menu (Option 1)

This menu is the same as the UNITS menu obtained by using ↱ UNITS . The applications of this menu are discussed in detail in Chapter 3.

## BASE convert menu (Option 2)

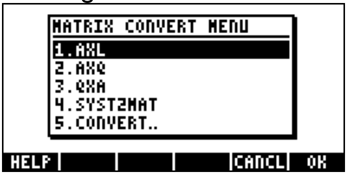
This menu is the same as the UNITS menu obtained by using ↱ BASE . The applications of this menu are discussed in detail in Chapter 19.

## TRIGONOMETRIC convert menu (Option 3)

This menu is the same as the TRIG menu obtained by using ↱ TRIG . The applications of this menu are discussed in detail in this Chapter.

## MATRICES convert menu (Option 5)

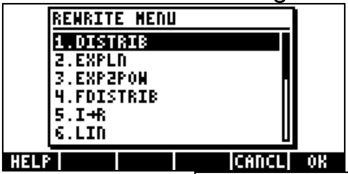
This menu contains the following functions:

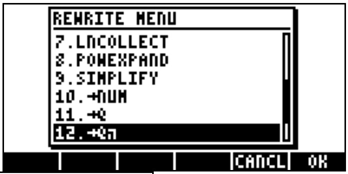


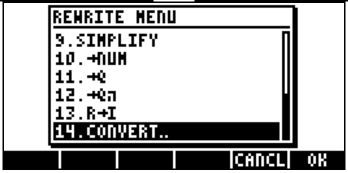
These functions are discussed in detail in Chapter 10.

## REWRITE convert menu (Option 4)

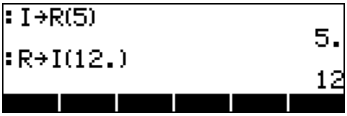
This menu contains the following functions:





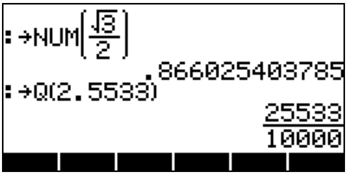


Functions I→R and R→I are used to convert a number from integer (I) to real (R), or vice versa. Integer numbers are shown without trailing decimal points, while real numbers representing integers will have a trailing decimal point, e.g.,



```
:I→R(5)
                    5.
:R→I(12.)
                    12
```

Function →NUM has the same effect as the keystroke combination ↱ →NUM (associated with the ENTER key). Function →NUM converts a symbolic result into its floating-point value. Function →Q converts a floating-point value into a fraction. Function →Qπ converts a floating-point value into a fraction of π, if a fraction of π can be found for the number; otherwise, it converts the number to a fraction. Examples are of these three functions are shown next.

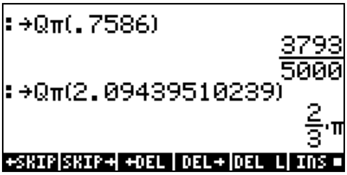


$:\to NUM\left(\frac{\sqrt{3}}{2}\right)$

.866025403785

$:\to Q(2.5533)$

$\frac{25533}{10000}$



$:\to Q\pi(.7586)$

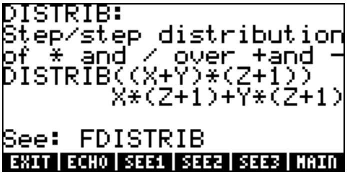
$\frac{3793}{5000}$

$:\to Q\pi(2.09439510239)$

$\frac{2}{3}\cdot\pi$

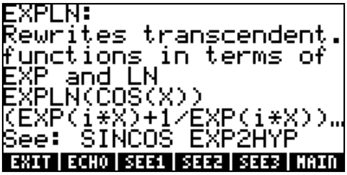
Out of the functions in the REWRITE menu, functions DISTRIB, EXPLN, EXP2POW, FDISTRIB, LIN, LNCOLLECT, POWEREXPAND, and SIMPLIFY apply to algebraic expressions. Many of these functions are presented in this Chapter. However, for the sake of completeness we present here the help-facility entries for these functions.

DISTRIB



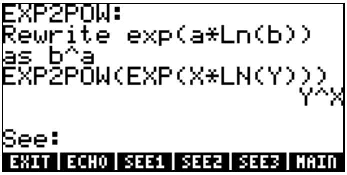
```
DISTRIB:
Step/step distribution
of * and / over +and -
DISTRIB((X+Y)*(Z+1))
       X*(Z+1)+Y*(Z+1)

See: FDISTRIB
```

EXPLN



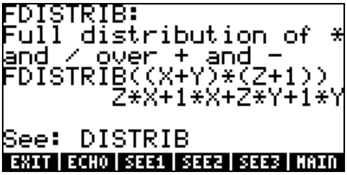
```
EXPLN:
Rewrites transcendent.
functions in terms of
EXP and LN
EXPLN(COS(X))
(EXP(i*X)+1/EXP(i*X))…
See: SINCOS EXP2HYP
```

EXP2POW



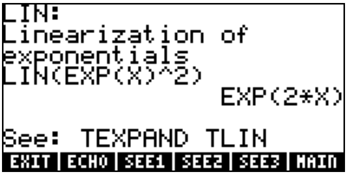
```
EXP2POW:
Rewrite exp(a*Ln(b))
as b^a
EXP2POW(EXP(X*LN(Y)))
                   Y^X

See:
```

FDISTRIB



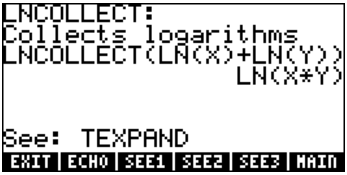
```
FDISTRIB:
Full distribution of *
and / over + and -
FDISTRIB((X+Y)*(Z+1))
       Z*X+1*X+Z*Y+1*Y

See: DISTRIB
```

LIN



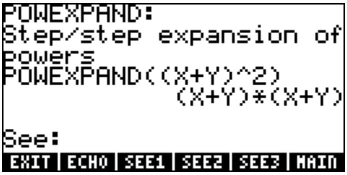
```
LIN:
Linearization of
exponentials
LIN(EXP(X)^2)
                 EXP(2*X)

See: TEXPAND TLIN
```

LNCOLLECT



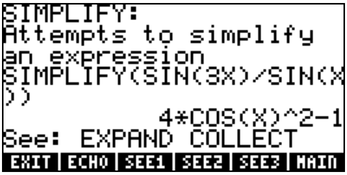
```
LNCOLLECT:
Collects logarithms
LNCOLLECT(LN(X)+LN(Y))
                  LN(X*Y)

See: TEXPAND
```

POWEREXPAND



```
POWEXPAND:
Step/step expansion of
powers
POWEXPAND((X+Y)^2)
             (X+Y)*(X+Y)

See:
```

SIMPLIFY



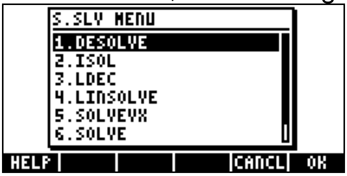
```
SIMPLIFY:
Attempts to simplify
an expression
SIMPLIFY(SIN(3X)/SIN(X
))
             4*COS(X)^2-1
See: EXPAND COLLECT
```

# Chapter 6
# Solution to single equations

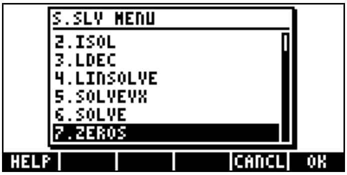
In this chapter we feature those functions that the calculator provides for solving single equations of the form f(X) = 0. Associated with the 7 key there are two menus of equation-solving functions, the Symbolic SOLVer (↰ S.SLV ), and the NUMerical SoLVer (↱ NUM.SLV ). Following, we present some of the functions contained in these menus. Change CAS mode to Complex for these exercises (see Chapter 2).

## Symbolic solution of algebraic equations

Here we describe some of the functions from the Symbolic Solver menu. Activate the menu by using the keystroke combination . With system flag 117 set to CHOOSE boxes, the following menu lists will be available:



```
S.SLV MENU
1.DESOLVE
2.ISOL
3.LDEC
4.LINSOLVE
5.SOLVEVX
6.SOLVE
HELP            CANCL  OK
```

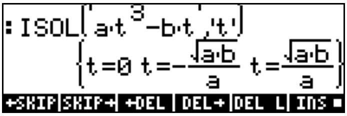


```
S.SLV MENU
2.ISOL
3.LDEC
4.LINSOLVE
5.SOLVEVX
6.SOLVE
7.ZEROS
HELP            CANCL  OK
```

Functions DESOLVE and LDEC are used for the solution of differential equations, the subject of a different chapter, and therefore will not be presented here. Similarly, function LINSOLVE relates to the solution of multiple linear equations, and it will be presented in a different chapter. Functions ISOL and SOLVE can be used to solve for any unknown in a polynomial equation. Function SOLVEVX solves a polynomial equation where the unknown is the default CAS variable VX (typically set to 'X'). Finally, function ZEROS provides the zeros, or roots, of a polynomial. Entries for all the functions in the S.SLV menu, except ISOL, are available through the CAS help facility (TOOL NXT HELP ).

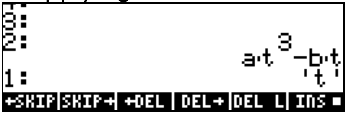
### Function ISOL

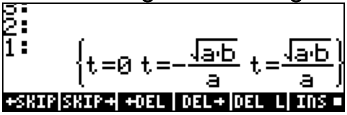
Function ISOL(Equation, variable) will produce the solution(s) to *Equation* by isolating *variable*. For example, with the calculator set to ALG mode, to solve for *t* in the equation $at^3$-bt = 0 we can use the following:


:ISOL('a·t^3-b·t','t')

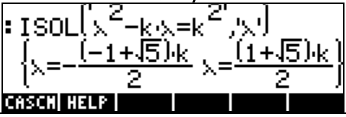
$$\left\{t=0\ \ t=-\frac{\sqrt{a\cdot b}}{a}\ \ t=\frac{\sqrt{a\cdot b}}{a}\right\}$$

Using the RPN mode, the solution is accomplished by entering the equation in the stack, followed by the variable, before entering function ISOL. Right before the execution of ISOL, the RPN stack should look as in the figure to the left. After applying ISOL, the result is shown in the figure to the right:


3:
2: $a\cdot t^3-b\cdot t$
1: 't'


3:
2:
1: $\left\{t=0\ \ t=-\frac{\sqrt{a\cdot b}}{a}\ \ t=\frac{\sqrt{a\cdot b}}{a}\right\}$

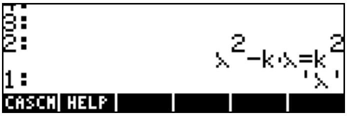
The first argument in ISOL can be an expression, as shown above, or an equation. For example, in ALG mode, try:

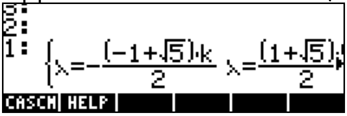

:ISOL('λ^2-k·λ=k^2','λ')

$$\left\{\lambda=-\frac{(-1+\sqrt{5})\cdot k}{2}\ \ \lambda=\frac{(1+\sqrt{5})\cdot k}{2}\right\}$$

**Note:** To type the equal sign (=) in an equation, use ↱ = (associated with the +/- key).

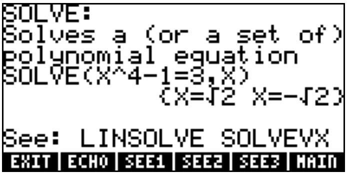
The same problem can be solved in RPN mode as illustrated below (figures show the RPN stack before and after the application of function ISOL):


3:
2: $\lambda^2-k\cdot\lambda=k^2$
1: 'λ'

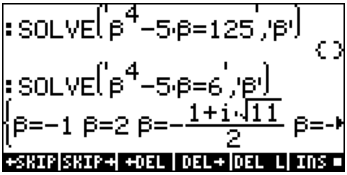

3:
2:
1: $\left\{\lambda=-\frac{(-1+\sqrt{5})\cdot k}{2}\ \ \lambda=\frac{(1+\sqrt{5})\cdot}{2}\right.$

## Function SOLVE

Function SOLVE has the same syntax as function ISOL, except that SOLVE can also be used to solve a set of polynomial equations. The help-facility entry for function SOLVE, with the solution to equation X^4 – 1 = 3 , is shown next:

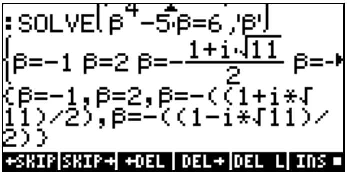

```
SOLVE:
Solves a (or a set of)
polynomial equation
SOLVE(X^4-1=3,X)
             {X=√2 X=-√2}

See: LINSOLVE SOLVEVX
```

The following examples show the use of function SOLVE in ALG and RPN modes:



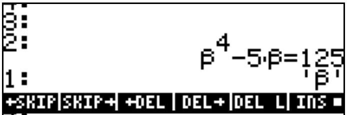
```
:SOLVE('β^4-5·β=125','β')
                              {}
:SOLVE('β^4-5·β=6','β')
{β=-1 β=2 β=-(1+i·√11)/2 β=-▸
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
```

The screen shot shown above displays two solutions. In the first one, β$^4$-5β =125, SOLVE produces no solutions { }. In the second one, β$^4$ - 5β = 6, SOLVE produces four solutions, shown in the last output line. The very last solution is not visible because the result occupies more characters than the width of the calculator's screen. However, you can still see all the solutions by using the down arrow key (▼), which triggers the line editor (this operation can be used to access any output line that is wider than the calculator's screen):

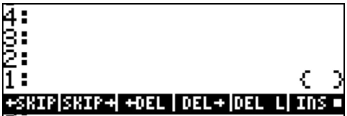


```
:SOLVE('β^4-5·β=6','β')
{β=-1 β=2 β=-(1+i·√11)/2 β=-▸
{β=-1,β=2,β=-((1+i*√(
11)/2),β=-((1-i*√11)/
2)}
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
```

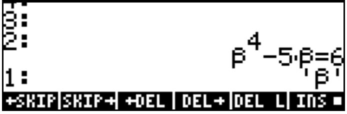
The corresponding RPN screens for these two examples, before and after the application of function SOLVE, are shown next:



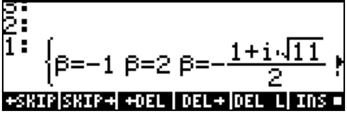
```
3:
2:            β^4-5·β=125
1:                    'β'
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
```



```
4:
3:
2:
1:                     { }
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
```

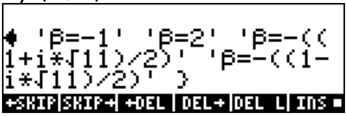


```
3:
2:              β^4-5·β=6
1:                    'β'
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
```



```
3:
2:
1: {β=-1 β=2 β=-(1+i·√11)/2 ▸
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
```

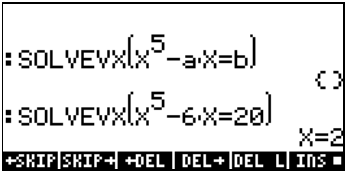
Use of the down-arrow key (▼) in this mode will launch the line editor:



```
{ 'β=-1' 'β=2' 'β=-((
1+i*√11)/2)' 'β=-((1-
i*√11)/2)' }
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
```

## Function SOLVEVX

The function SOLVEVX solves an equation for the default CAS variable contained in the reserved variable name VX. By default, this variable is set to 'X'. Examples, using the ALG mode with VX = 'X', are shown below:



:SOLVEVX$(X^5-a\cdot X=b)$

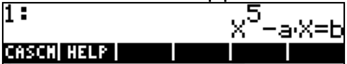
{ }

:SOLVEVX$(X^5-6\cdot X=20)$

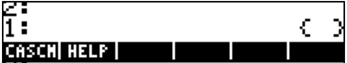
X=2

In the first case SOLVEVX could not find a solution. In the second case, SOLVEVX found a single solution, X = 2.

The following screens show the RPN stack for solving the two examples shown above (before and after application of SOLVEVX):

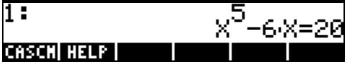


1: $X^5-a\cdot X=b$

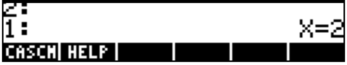


2:

1: { }



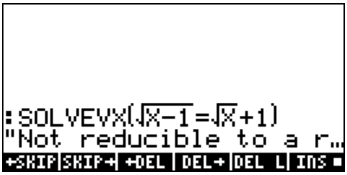
1: $X^5-6\cdot X=20$



2:

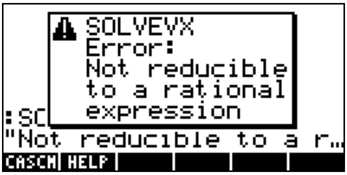
1: X=2

The equation used as argument for function SOLVEVX must be reducible to a rational expression. For example, the following equation will not processed by SOLVEVX:



:SOLVEVX$(\sqrt{X-1}=\sqrt{X}+1)$

"Not reducible to a r…



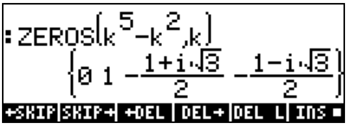
SOLVEVX Error: Not reducible to a rational expression

:SC

"Not reducible to a r…

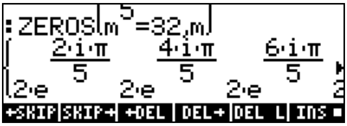
## Function ZEROS

The function ZEROS finds the solutions of a polynomial equation, without showing their multiplicity. The function requires having as input the expression for the equation and the name of the variable to solve for. Examples in ALG mode are shown next:



:ZEROS$(k^5-k^2,k)$

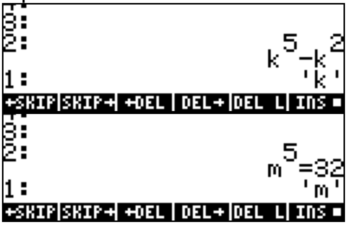
$\left\{0\ 1\ -\frac{1+i\cdot\sqrt{3}}{2}\ -\frac{1-i\cdot\sqrt{3}}{2}\right\}$



:ZEROS$(m^5=32,m)$

$\left\{2\cdot e^{\frac{2\cdot i\cdot\pi}{5}}\ 2\cdot e^{\frac{4\cdot i\cdot\pi}{5}}\ 2\cdot e^{\frac{6\cdot i\cdot\pi}{5}}\ 2\right.$

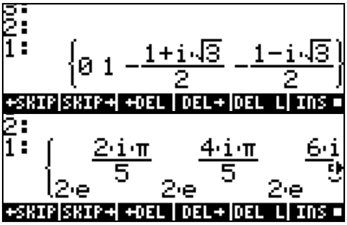
To use function ZEROS in RPN mode, enter first the polynomial expression, then the variable to solve for, and then function ZEROS. The following screen shots show the RPN stack before and after the application of ZEROS to the two examples above:



3:
2: $k^5-k^2$
1: 'k'

3:
2: $m^5=32$
1: 'm'

←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■



3:
2:
1: $\left\{0\ 1\ -\frac{1+i\cdot\sqrt{3}}{2}\ -\frac{1-i\cdot\sqrt{3}}{2}\right\}$

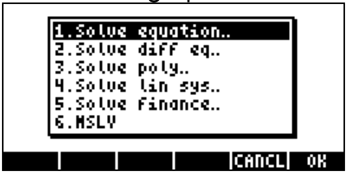
2:
1: $\left\{2\cdot e^{\frac{2\cdot i\cdot\pi}{5}}\ 2\cdot e^{\frac{4\cdot i\cdot\pi}{5}}\ 2\cdot e^{\frac{6\cdot i}{5}}\right.$

←SKIP SKIP→ ←DEL DEL→ DEL L INS ■
←SKIP SKIP→ ←DEL DEL→ DEL L INS ■

The Symbolic Solver functions presented above produce solutions to rational equations (mainly, polynomial equations). If the equation to be solved for has all numerical coefficients, a numerical solution is possible through the use of the Numerical Solver features of the calculator.

## Numerical solver menu

The calculator provides a very powerful environment for the solution of single algebraic or transcendental equations. To access this environment we start the numerical solver (NUM.SLV) by using ↱ NUM.SLV . This produces a drop-down menu that includes the following options:



Item *2. Solve diff eq..* is to be discussed in a later chapter on differential equations. Item *4. Solve lin sys..* will be discussed in a later Chapter on matrices. Item *6. MSLV* (Multiple equation SoLVer) will be presented in the next chapter. Following, we present applications of items *3. Solve poly..*, *5. Solve finance*, and *1. Solve equation..*, in that order. Appendix 1-A, at the end of Chapter 1, contains instructions on how to use input forms with examples for the numerical solver applications.

**Notes:**
1. Whenever you solve for a value in the NUM.SLV applications, the value solved for will be placed in the stack. This is useful if you need to keep that value available for other operations.
2. There will be one or more variables created whenever you activate some of the applications in the NUM.SLV menu.

## Polynomial Equations

Using the *Solve poly…* option in the calculator's *SOLVE* environment you can:
(1) find the solutions to a polynomial equation;
(2) obtain the coefficients of the polynomial having a number of given roots;
(3) obtain an algebraic expression for the polynomial as a function of X.

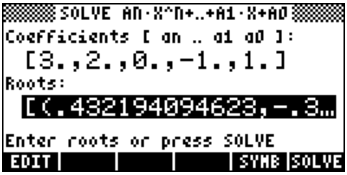
### Finding the solutions to a polynomial equation

A polynomial equation is an equation of the form: $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0$. The fundamental theorem of algebra indicates that there are *n* solutions to any polynomial equation of order *n*. Some of the solutions could be complex numbers, nevertheless. As an example, solve the equation: $3s^4 + 2s^3 - s + 1 = 0$.

We want to place the coefficients of the equation in a vector [$a_n$,$a_{n-1}$,$a_1$ $a_0$]. For this example, let's use the vector [3,2,0,-1,1]. To solve for this polynomial equation using the calculator, try the following:

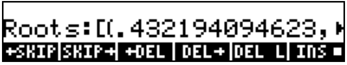
| | |
|---|---|
| ↱ NUM.SLV ▼ ▼ OK | Select Solve poly… |
| ↰ [] 3 ↱ , 2 ↱ , 0 | |
| ↱ , 1 +/- ↱ , 1 OK | Enter vector of coefficients |
| SOLVE | Solve equation |

The screen will show the solution as follows:

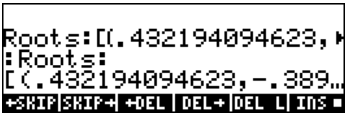


```
SOLVE AN·X^N+..+A1·X+A0
Coefficients [ an .. a1 a0 ]:
 [3.,2.,0.,-1.,1.]
Roots:
 [(.432194094623,-.3…
Enter roots or press SOLVE
```

Press ENTER to return to stack. The stack will show the following results in ALG mode (the same result would be shown in RPN mode):


Roots:[(.432194094623,▸

To see all the solutions, press the down-arrow key (▼) to trigger the line editor:


Roots:[(.432194094623,▸
:Roots:
[(.432194094623,-.389…

All the solutions are complex numbers: (0.432,-0.389), (0.432,0.389), (-0.766, 0.632), (-0.766, -0.632).

**Note**: Recall that complex numbers in the calculator are represented as ordered pairs, with the first number in the pair being the real part, and the second number, the imaginary part. For example, the number (0.432,-0.389), a complex number, will be written normally as 0.432 - 0.389*i*, where *i* is the imaginary unit, i.e., $i^2 = -1$.

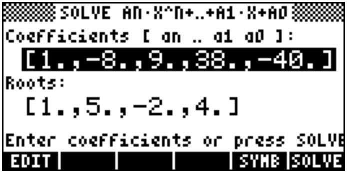
**Note**: The *fundamental theorem of algebra* indicates that there are *n* solutions for any polynomial equation of order *n*. There is another theorem of algebra that indicates that if one of the solutions to a polynomial equation with real coefficients is a complex number, then the conjugate of that number is also a solution. In other words, complex solutions to a polynomial equation with real coefficients come in pairs. That means that polynomial equations with real coefficients of odd order will have at least one real solution.

### Generating polynomial coefficients given the polynomial's roots

Suppose you want to generate the polynomial whose roots are the numbers [1, 5, -2, 4]. To use the calculator for this purpose, follow these steps:

| | |
|---|---|
| → NUM.SLV ▼ ▼ OK | Select Solve poly… |
| ▼ ← [] 1 → , 5 | |
| → , 2 +/- → , 4 OK | Enter vector of roots |
| SOLVE | Solve for coefficients |

Press ENTER to return to stack, the coefficients will be shown in the stack.



```
SOLVE AN·X^N+..+A1·X+A0
Coefficients [ an .. a1 a0 ]:
 [1.,-8.,9.,38.,-40.]
Roots:
 [1.,5.,-2.,4.]
Enter coefficients or press SOLVE
EDIT |    |    |    |SYMB|SOLVE
```

Press ▼ to trigger the line editor to see all the coefficients.

**Note**: If you want to get a polynomial with real coefficients, but having complex roots, you must include the complex roots in pairs of conjugate numbers. To illustrate the point, generate a polynomial having the roots [1 (1,2) (1,-2)]. Verify that the resulting polynomial has only real coefficients. Also, try generating a polynomial with roots [1 (1,2) (-1,2)], and verify that the resulting polynomial will have complex coefficients.

### Generating an algebraic expression for the polynomial

You can use the calculator to generate an algebraic expression for a polynomial given the coefficients or the roots of the polynomial. The resulting expression will be given in terms of the default CAS variable X. (The examples below shows how you can replace X with any other variable by using the function |.)

To generate the algebraic expression using the coefficients, try the following example. Assume that the polynomial coefficients are [1,5,-2,4]. Use the following keystrokes:

| | |
|---|---|
| ↱ NUM.SLV ▼ ▼ OK | Select Solve poly… |
| ↰ [] 1 ↱ , 5 | Enter vector of coefficients |
| ↱ , 2 +/- ↱ , 4 OK | |
| ▲ SYMB | Generate symbolic expression |
| ENTER | Return to stack. |

The expression thus generated is shown in the stack as:

'X^3+5*X^2+-2*X+4'.

To generate the algebraic expression using the roots, try the following example. Assume that the polynomial roots are [1,3,-2,1]. Use the following keystrokes:

| | |
|---|---|
| ↱ NUM.SLV ▼ ▼ OK | Select Solve poly… |
| ▼ ↰ [] 1 ↱ , 3 | Enter vector of roots |
| ↱ , 2 +/- ↱ , 1 OK | |
| ▼ SYMB | Generate symbolic expression |
| ENTER | Return to stack. |

The expression thus generated is shown in the stack as:'(X-1)*(X-3)*(X+2)*(X-1)'. To expand the products, you can use the EXPAND command. The resulting expression is: 'X^4+-3*X^3+ -3*X^2+11*X-6'.

A different approach to obtaining an expression for the polynomial is to generate the coefficients first, then generate the algebraic expression with the coefficients highlighted. For example, for this case try:

| | |
|---|---|
| ↱ NUM.SLV ▼ ▼ OK | Select Solve poly… |
| ▼ ↰ [] 1 ↱ , 3 | Enter vector of roots |
| ↱ , 2 +/- ↱ , 1 OK | |
| SOLVE | Solve for coefficients |
| ▼ SYMB | Generate symbolic expression |
| ENTER | Return to stack. |

The expression thus generated is shown in the stack as: 'X^4+-3*X^3+ -3*X^2+11*X+-6*X^0'. The coefficients are listed in stack level 2.

## Financial calculations

The calculations in item *5. Solve finance..* in the Numerical Solver (*NUM.SLV*) are used for calculations of time value of money of interest in the discipline of engineering economics and other financial applications. This application can also be started by using the keystroke combination ↰ $ FINANCE (associated with the 9 key). Before discussing in detail the operation of this solving environment, we present some definitions needed to understand financial operations in the calculator.

### Definitions

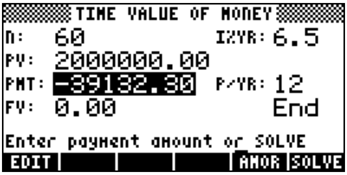
Often, to develop projects, it is necessary to borrow money from a financial institution or from public funds. The amount of money borrowed is referred to as the *Present Value* (PV). This money is to be repaid through *n* periods (typically multiples or sub-multiples of a month) subject to an *annual interest rate* of I%YR. The *number of periods per year* (P/YR) is an integer number of periods in which the year will be divided for the purpose of repaying the loan money. Typical values of P/YR are 12 (one payment per month), 24 (payment twice a month), or 52 (weekly payments). The *payment*(PMT) is the amount that the borrower must pay to the lender at the beginning or end of each of the *n* periods of the loan. The *future value* of the money (FV) is the value that the borrowed amount of money will be worth at the end of *n* periods. Typically payment occurs at the end of each period, so that the borrower starts paying at the end of the first period, and pays the same fixed amount at the end of the second, third, etc., up to the end of the *n*-th period.

### Example 1 – Calculating payment on a loan

If $2 million are borrowed at an annual interest rate of 6.5% to be repaid in 60 monthly payments, what should be the monthly payment? For the debt to be totally repaid in 60 months, the future values of the loan should be zero. So, for the purpose of using the financial calculation feature of the calculator we will use the following values: n = 60, I%YR = 6.5, PV = 2000000, FV = 0, P/YR = 12. To enter the data and solve for the payment, PMT, use:

| | |
|---|---|
| ↰ FINANCE | Start the financial calculation input form |
| 60 OK | Enter n = 60 |
| 6.5 OK | Enter I%YR = 6.5 % |
| 2000000 OK | Enter PV = 2,000,000 US$ |
| ▼ | Skip PMT, since we will be solving for it |
| 0 OK | Enter FV = 0, the option End is highlighted |
| ▲ ◄ SOLVE | Highlight PMT and solve for it |

The solution screen will look like this:

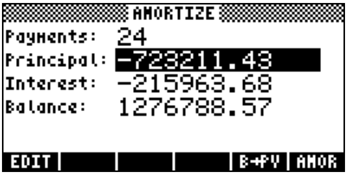


```
TIME VALUE OF MONEY
n:   60                I%YR: 6.5
PV:  2000000.00
PMT: -89132.30         P/YR: 12
FV:  0.00                    End

Enter payment amount or SOLVE
EDIT |     |     |     | AMOR |SOLVE
```

The screen now shows the value of PMT as –39,132.30, i.e., the borrower must pay the lender US $ 39,132.30 at the end of each month for the next 60 months to repay the entire amount. The reason why the value of PMT turned out to be negative is because the calculator is looking at the money amounts from the point of view of the borrower. The borrower has + US $ 2,000,000.00 at time period t = 0, then he starts paying, i.e., adding -US $ 39132.30 at times t = 1, 2, …, 60. At t = 60, the net value in the hands of the borrower is zero. Now, if you take the value US $ 39,132.30 and multiply it by the 60 payments, the total paid back by the borrower is US $ 2,347,937.79. Thus, the lender makes a net profit of $ 347,937.79 in the 5 years that his money is used to finance the borrower's project.

### Example 2 – Calculating amortization of a loan

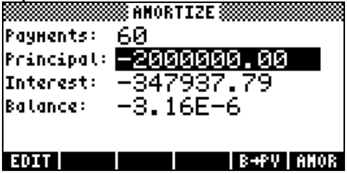
The same solution to the problem in Example 1 can be found by pressing AMOR, which is stands for AMORTIZATION. This option is used to calculate how much of the loan has been amortized at the end of a certain number of payments. Suppose that we use 24 periods in the first line of the amortization screen, i.e., 2 4 OK. Then, press AMOR. You will get the following result:



```
AMORTIZE
Payments:  24
Principal: -723211.43
Interest:  -215963.68
Balance:   1276788.57
```

This screen is interpreted as indicating that after 24 months of paying back the debt, the borrower has paid up US $ 723,211.43 into the principal amount borrowed, and US $ 215,963.68 of interest. The borrower still has to pay a balance of US $1,276,788.57 in the next 36 months.

Check what happens if you replace 60 in the *Payments:* entry in the amortization screen, then press OK AMOR. The screen now looks like this:



```
AMORTIZE
Payments:  60
Principal: -2000000.00
Interest:  -347937.79
Balance:   -3.16E-6
```

This means that at the end of 60 months the US $ 2,000,000.00 principal amount has been paid, together with US $ 347,937.79 of interest, with the balance being that the lender owes the borrower US $ 0.000316. Of course, the balance should be zero. The value shown in the screen above is simply round-off error resulting from the numerical solution.

Press ON or ENTER, twice, to return to normal calculator display.

**Example 3 – Calculating payment with payments at beginning of period**
Let's solve the same problem as in Examples 1 and 2, but using the option that payment occurs at the beginning of the payment period. Use:

| | |
|---|---|
| ← FINANCE | Start the financial calculation input form |
| 60 OK | Enter n = 60 |
| 6.5 OK | Enter I%YR = 6.5 % |
| 2000000 OK | Enter PV = 2,000,000 US$ |
| ▼ | Skip PMT, since we will be solving for it |
| 0 OK | Enter FV = 0, the option End is highlighted |
| CHOOS ▲ OK | Change payment option to *Begin* |
| ▲ ◄ SOLVE | Highlight PMT and solve for it |

The screen now shows the value of PMT as –38,921.47, i.e., the borrower must pay the lender US $ 38,921.48 at the *beginning* of each month for the next 60 months to repay the entire amount. Notice that the amount the borrower pays monthly, if paying at the beginning of each payment period, is slightly smaller than that paid at the end of each payment period. The reason for that difference is that the lender gets interest earnings from the payments from the beginning of the period, thus alleviating the burden on the lender.

**Notes:**
1. The financial calculator environment allows you to solve for any of the terms involved, i.e., n, I%YR, PV, FV, P/Y, given the remaining terms in the loan calculation. Just highlight the value you want to solve for, and press SOLVE. The result will be shown in the highlighted field.

2. The values calculated in the financial calculator environment are copied to the stack with their corresponding tag (identifying label).

### Deleting the variables

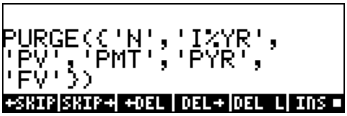
When you use the financial calculator environment for the first time within the HOME directory, or any sub-directory, it will generate the variables n I%YR PV PMT PYR FV to store the corresponding terms in the calculations.. You can see the contents of these variables by using:

↱ n ↱ I%YR ↱ PV ↱ PMT ↱ PYR ↱ FV.

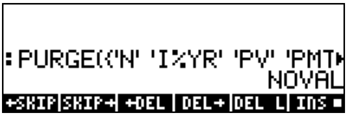
You can either keep these variables for future use, or use the PURGE function to erase them from your directory. <u>To erase all of the variables at once</u>, if using ALG mode, try the following:

| | |
|---|---|
| TOOL PURGE VAR ↰ {} | Enter PURGE, prepare list of variables |
| ' ↱ n | Enter name of variable N |
| ▶ ↱ , | Enter a comma |
| ' ↱ I%YR | Enter name of variable I%YR |
| ▶ ↱ , | Enter a comma |
| ' ↱ PV | Enter name of variable PV |
| ▶ ↱ , | Enter a comma |
| ' ↱ PMT | Enter name of variable PMT |
| ▶ ↱ , | Enter a comma |
| ' ↱ PYR | Enter name of variable PYR |
| ▶ ↱ , | Enter a comma |
| ' ↱ FV. | Enter name of variable FV |
| ENTER | Execute PURGE command |

The following two screen shots show the PURGE command for purging all the variables in the directory, and the result after executing the command.



```
PURGE({'N','I%YR',
'PV','PMT','PYR',
'FV'})
```

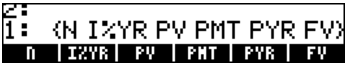


```
:PURGE({'N' 'I%YR' 'PV' 'PMT▸
                         NOVAL
```

In RPN mode, the command is executed by using:

| | |
|---|---|
| VAR ← {} | Prepare a list of variables to be purged |
| n | Enter name of variable N |
| I%YR | Enter name of variable I%YR |
| PV | Enter name of variable PV |
| PMT | Enter name of variable PMT |
| PYR | Enter name of variable PYR |
| FV | Enter name of variable FV |
| ENTER | Enter list of variables in stack |
| TOOL PURGE | Purge variables in list |

Before the command PURGE is entered, the RPN stack will look like this:



```
2:
1:  {N I%YR PV PMT PYR FV}
```

## Solving equations with one unknown through NUM.SLV

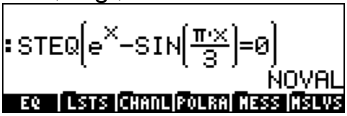
The calculator's NUM.SLV menu provides item *1. Solve equation..* solve different types of equations in a single variable, including non-linear algebraic and transcendental equations. For example, let's solve the equation: $e^x$-*sin(πx/3) = 0.*

Simply enter the expression as an algebraic object and store it into variable EQ. The required keystrokes in ALG mode are the following:

' ← e^x ALPHA ← X ▶ − SIN ← π
× ALPHA ← X ÷ 3 ▶ → = 0 ▶
STO▶ ALPHA E ALPHA Q ENTER

**Function STEQ**

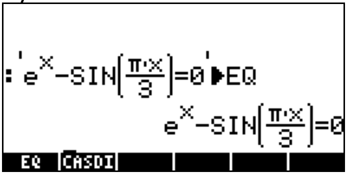
Function STEQ, available through the command catalog, → CAT , will store its argument into variable EQ, e.g., in ALG mode:



```
:STEQ(e^X-SIN(π·X/3)=0)
                       NOVAL
```

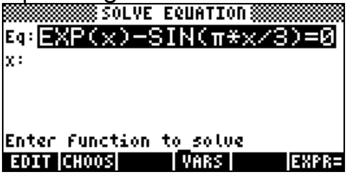
In RPN mode, enter the equation between apostrophes and activate command STEQ. Thus, function STEQ can be used as a shortcut to store an expression into variable EQ.

Press VAR to see the newly created EQ variable:



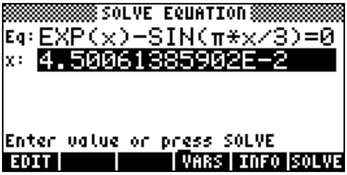
```
:'e^X-SIN(π·x/3)=0'▶EQ
                  e^X-SIN(π·x/3)=0
```

Then, enter the SOLVE environment and select *Solve equation…*, by using: ↱ NUM.SLV OK. The corresponding screen will be shown as:



```
SOLVE EQUATION
Eq: EXP(x)-SIN(π*x/3)=0
x:

Enter function to solve
```

The equation we stored in variable EQ is already loaded in the *Eq* field in the SOLVE EQUATION input form. Also, a field labeled *x* is provided. To solve the equation all you need to do is highlight the field in front of X: by using ▽, and press SOLVE. The solution shown is X: 4.5006E-2:



```
SOLVE EQUATION
Eq: EXP(x)-SIN(π*x/3)=0
x: 4.50061385902E-2

Enter value or press SOLVE
```

This, however, is not the only possible solution for this equation. To obtain a negative solution, for example, enter a negative number in the X: field before solving the equation. Try 3 +/- OK ▽ SOLVE. The solution is now X: -3.045.

### Solution procedure for *Equation Solve...*

The numerical solver for single-unknown equations works as follows:

- It lets the user type in or CHOOSe an equation to solve.
- It creates an input form with input fields corresponding to all variables involved in equation stored in variable EQ.
- The user needs to enter values for all variables involved, except one.

- The user then highlights the field corresponding to the unknown for which to solve the equation, and presses SOLVE
- The user may force a solution by providing an initial guess for the solution in the appropriate input field before solving the equation.

The calculator uses a search algorithm to pinpoint an interval for which the function changes sign, which indicates the existence of a root or solution. It then utilizes a numerical method to converge into the solution.

The solution the calculator seeks is determined by the initial value present in the unknown input field. If no value is present, the calculator uses a default value of zero. Thus, you can search for more than one solution to an equation by changing the initial value in the unknown input field. Examples of the equations solutions are shown following.

**Example 1 – Hooke's law for stress and strain**

The equation to use is Hooke's law for the normal strain in the x-direction for a solid particle subjected to a state of stress given by

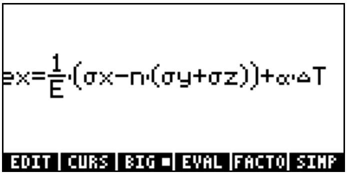
$$\begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}$$

The equation is $e_{xx} = \frac{1}{E}[\sigma_{xx} - n \cdot (\sigma_{yy} + \sigma_{zz})] + \alpha \cdot \Delta T,$ here $e_{xx}$ is the unit strain in the x-direction, $\sigma_{xx}$, $\sigma_{yy}$, and $\sigma_{zz}$, are the normal stresses on the particle in the directions of the x-, y-, and z-axes, *E* is Young's modulus or modulus of elasticity of the material, *n* is the Poisson ratio of the material, $\alpha$ is the thermal expansion coefficient of the material, and $\Delta T$ is a temperature increase.

Suppose that you are given the following data: $\sigma_{xx}$= 2500 psi, $\sigma_{yy}$ =1200 psi, and $\sigma_{zz}$ = 500 psi, E = 1200000 psi, *n* = 0.15, α = 0.00001/°F, ΔT = 60 °F. To calculate the strain $e_{xx}$ use the following:

| | |
|---|---|
| ↱ NUM.SLV OK | Access numerical solver to solve equations |
| ↱ EQW | Access the equation writer to enter equation |

At this point follow the instructions from Chapter 2 on how to use the Equation Writer to build an equation. The equation to enter in the *Eq* field should look like this (notice that we use only one sub-index to refer to the variables, i.e., $e_{xx}$ is translated as *ex*, etc. -- this is done to save typing time):



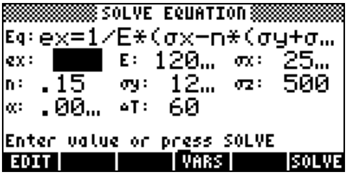
$ex=\frac{1}{E}\cdot(\sigma x-n\cdot(\sigma y+\sigma z))+\alpha\cdot\Delta T$

Use the following shortcuts for special characters:

σ: ALPHA ↱ S    α: ALPHA ↱ A    Δ: ALPHA ↱ C

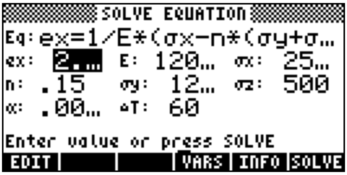
and recall that lower-case letters are entered by using ALPHA ↰ before the letter key, thus, x is typed as ALPHA ↰ X .

Press ENTER to return to the solver screen. Enter the values proposed above into the corresponding fields, so that the solver screen looks like this:



```
SOLVE EQUATION
Eq: ex=1/E*(σx−n*(σy+σ…
ex: ■      E: 120…   σx: 25…
n: .15     σy: 12…   σz: 500
α: .00…    ΔT: 60
Enter value or press SOLVE
```

With the *ex:* field highlighted, press SOLVE to solve for *ex*:

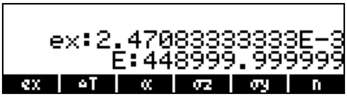


```
SOLVE EQUATION
Eq: ex=1/E*(σx−n*(σy+σ…
ex: 2.…    E: 120…   σx: 25…
n: .15     σy: 12…   σz: 500
α: .00…    ΔT: 60
Enter value or press SOLVE
```

The solution can be seen from within the SOLVE EQUATION input form by pressing EDIT while the *ex:* field is highlighted. The resulting value is 2.470833333333E-3. Press OK to exit the EDIT feature.

Suppose that you now, want to determine the Young's modulus that will produce a strain of $e_{xx}$ = 0.005 under the same state of stress, neglecting thermal expansion. In this case, you should enter a value of 0.005 in the *ex:* field, and a zero in the *ΔT:* field (with ΔT = 0, no thermal effects are included). To solve for E, highlight the *E:* field and press SOLVE. The result, seeing with the EDIT feature is, E ≈449000 psi. Press SOLVE ENTER to return to normal display.

Notice that the results of the calculations performed within the numerical solver environment have been copied to the stack:



```
   ex:2.47083333333E-3
       E:448999.999999
```

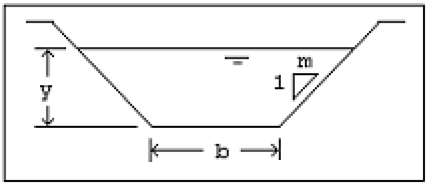
Also, you will see in your soft-menu key labels variables corresponding to those variables in the equation stored in EQ (press NXT to see all variables in your directory), i.e., variables *ex, ΔT, α, σz, σy, n, σx,* and *E*.

**Example 2 – Specific energy in open channel flow**

Specific energy in an open channel is defined as the energy per unit weight measured with respect to the channel bottom. Let E = specific energy, y = channel depth, V = flow velocity, g = acceleration of gravity, then we write

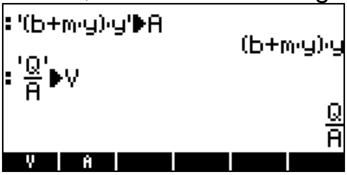
$$E = y + \frac{V^2}{2g}.$$

The flow velocity, in turn, is given by V = Q/A, where Q = water discharge, A = cross-sectional area. The area depends on the cross-section used, for example, for a trapezoidal cross-section, as shown in the figure below, A = (b+m·y) ·y, where b = bottom width, and m = side slope of cross section.

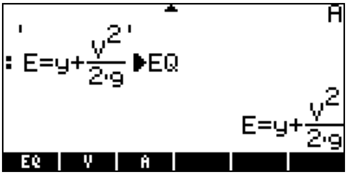


We can type in the equation for E as shown above and use auxiliary variables for A and V, so that the resulting input form will have fields for the fundamental variables y, Q, g, m, and b, as follows:

- First, create a sub-directory called SPEN (SPecific ENergy) and work within that sub-directory.
- Next, define the following variables:

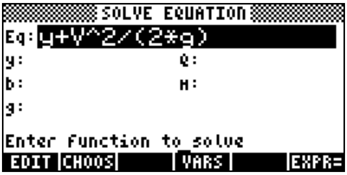


```
:'(b+m·y)·y'▶A
                    (b+m·y)·y
:'Q/A'▶V
                          Q/A
```



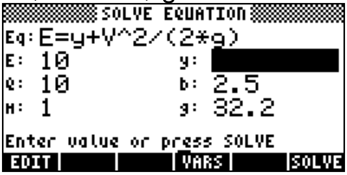
```
                              A
:'E=y+V^2/(2·g)'▶EQ
                 E=y+V^2/(2·g)
```

- Launch the numerical solver for solving equations: ↱ NUM.SLV OK. Notice that the input form contains entries for the variables y, Q, b, m, and g:



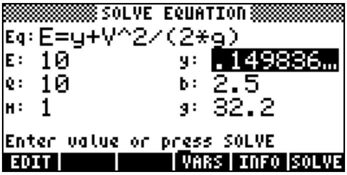
```
SOLVE EQUATION
Eq: y+V^2/(2*g)
y:            Q:
b:            m:
g:
Enter function to solve
```

- Try the following input data: E = 10 ft, Q = 10 cfs (cubic feet per second), b = 2.5 ft, m = 1.0, g = 32.2 ft/s$^2$:



```
SOLVE EQUATION
Eq: E=y+V^2/(2*g)
E: 10         y:
Q: 10         b: 2.5
m: 1          g: 32.2
Enter value or press SOLVE
```

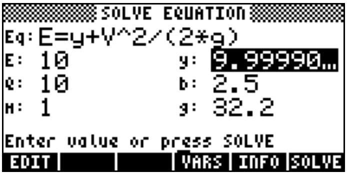
- Solve for y.



```
SOLVE EQUATION
Eq: E=y+V^2/(2*g)
E: 10          y: .149836…
Q: 10          b: 2.5
H: 1           g: 32.2
Enter value or press SOLVE
```

The result is 0.149836.., i.e., y = 0.149836.

- It is known, however, that there are actually two solutions available for y in the specific energy equation. The solution we just found corresponds to a numerical solution with an initial value of 0 (the default value for *y*, i.e., whenever the solution field is empty, the initial value is zero). To find the other solution, we need to enter a larger value of y, say 15, highlight the y input field and solve for y once more:



```
SOLVE EQUATION
Eq: E=y+V^2/(2*g)
E: 10          y: 9.99990…
Q: 10          b: 2.5
H: 1           g: 32.2
Enter value or press SOLVE
```

The result is now 9.99990, i.e., y = 9.99990 ft.

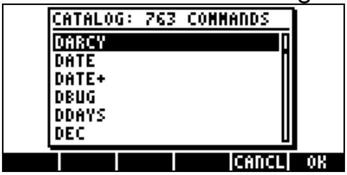
This example illustrates the use of auxiliary variables to write complicated equations. When NUM.SLV is activated, the substitutions implied by the auxiliary variables are implemented, and the input screen for the equation provides input field for the primitive or fundamental variables resulting from the substitutions. The example also illustrates an equation that has more than one solution, and how choosing the initial guess for the solution may produce those different solutions.

In the next example we will use the DARCY function for finding friction factors in pipelines. Thus, we define the function in the following frame.

**Special function for pipe flow: DARCY (ε/D,Re)**

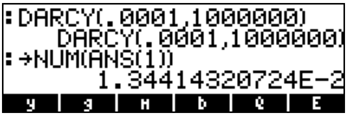
The Darcy-Weisbach equation is used to calculate the energy loss (per unit weight), $h_f$, in a pipe flow through a pipe of diameter D, absolute roughness ε, and length L, when the flow velocity in the pipe is V. The equation is written as $h_f = f \cdot \frac{L}{D} \cdot \frac{V^2}{2g}$. The quantity *f* is known as the friction factor of the flow and it has been found to be a function of the relative roughness of the pipe, /D, and a (dimensionless) Reynolds number, Re. The Reynolds number is defined as Re = ρVD/μ = VD/ν, where ρ and μ are the density and dynamic viscosity of the fluid, respectively, and ν = μ/ρ is the kinematic viscosity of the fluid.

The calculator provides a function called DARCY that uses as input the relative roughness ε/D and the Reynolds number, in that order, to calculate the friction factor f. The function DARCY can be found through the command catalog:



```
CATALOG: 763 COMMANDS
DARCY
DATE
DATE+
DBUG
DDAYS
DEC
```

For example, for e/D = 0.0001, Re = 1000000, you can find the friction factor by using: DARCY(0.0001,1000000). In the following screen, the function →NUM () was used to obtain a numerical value of the function:

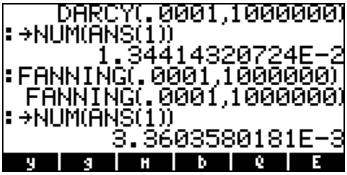


```
:DARCY(.0001,1000000)
    DARCY(.0001,1000000)
:→NUM(ANS(1))
       1.34414320724E-2
```

The result is f = DARCY(0.0001,1000000) = 0.01341…

### The function FANNING(ε/D,Re)

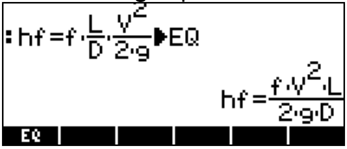
In aerodynamics applications a different friction factor, the Fanning friction factor, is used. The Fanning friction factor, $f_F$, is defined as 4 times the Darcy-Weisbach friction factor, *f*. The calculator also provides a function called FANNING that uses the same input as DARCY, i.e., e/D and Re, and provides the FANNING friction factor. Check that FANNING(0.0001,1000000) = 0.0033603589181s.



```
   DARCY(.0001,1000000)
:→NUM(ANS(1))
      1.34414320724E-2
:FANNING(.0001,1000000)
 FANNING(.0001,1000000)
:→NUM(ANS(1))
       3.3603580181E-3
```

### Example 3 – Flow in a pipe

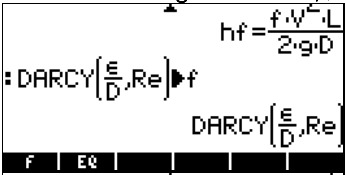
You may want to create a separate sub-directory (PIPES) to try this example. The main equation governing flow in a pipe is, of course, the Darcy-Weisbach equation. Thus, type in the following equation into EQ:



:$hf=f\cdot\frac{L}{D}\cdot\frac{V^2}{2\cdot g}$▶EQ

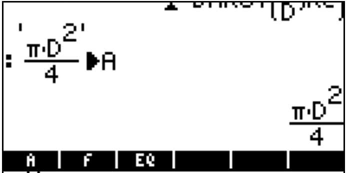
$hf=\frac{f\cdot V^2\cdot L}{2\cdot g\cdot D}$

Also, enter the following variables (f, A, V, Re):



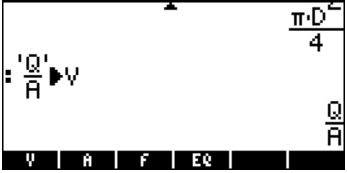
:DARCY$\left(\frac{\varepsilon}{D},Re\right)$▶f

DARCY$\left(\frac{\varepsilon}{D},Re\right)$



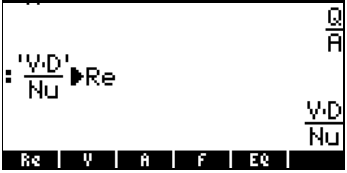
:$'\frac{\pi\cdot D^2}{4}'$▶A

$\frac{\pi\cdot D^2}{4}$



:$'\frac{Q}{A}'$▶V

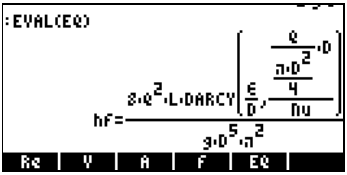
$\frac{Q}{A}$



:$'\frac{V\cdot D}{Nu}'$▶Re

$\frac{V\cdot D}{Nu}$

In this case we stored the main equation (Darcy-Weisbach equation) into EQ, and then replaced several of its variables by other expressions through the definition of variables f, A, V, and Re. To see the combined equation, use EVAL(EQ). In this example we changed the display setting so that we can see the entire equation in the screen:



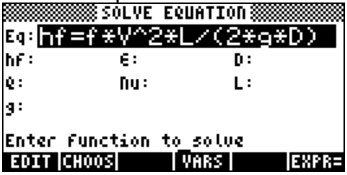
:EVAL(EQ)

$hf=\frac{8\cdot Q^2\cdot L\cdot DARCY\left(\frac{\varepsilon}{D},\frac{\frac{Q}{\frac{\pi\cdot D^2}{4}}\cdot D}{Nu}\right)}{g\cdot D^5\cdot\pi^2}$

Thus, the equation we are solving, after combining the different variables in the directory, is:

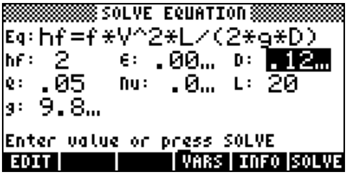
$$h_f = \frac{8Q^2 L}{\pi^2 g D^5} \cdot DARCY\left( \frac{\varepsilon}{D}, \frac{\frac{QD}{\pi D^2/4}}{Nu} \right)$$

The combined equation has primitive variables: $h_f$, $Q$, $L$, $g$, $D$, $\varepsilon$, and $Nu$. Launch the numerical solver (↱ NUM.SLV OK) to see the primitive variables listed in the SOLVE EQUATION input form:



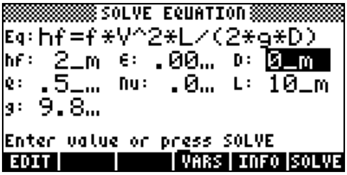
```
SOLVE EQUATION
Eq: hf=f*V^2*L/(2*g*D)
hf:        ε:        D:
Q:         Nu:       L:
g:
Enter function to solve
EDIT CHOOS VARS EXPR=
```

Suppose that we use the values hf = 2 m, ε = 0.00001 m, Q = 0.05 $m^3/s$, Nu = 0.000001 $m^2/s$, L = 20 m, and g = 9.806 $m/s^2$, find the diameter D. Enter the input values, and solve for D, The solution is: 0.12, i.e., D = 0.12 m.

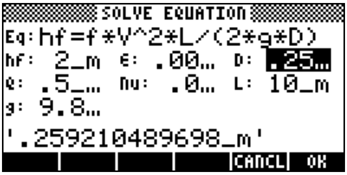


```
SOLVE EQUATION
Eq: hf=f*V^2*L/(2*g*D)
hf: 2    ε: .00…  D: .12…
Q: .05   Nu: .0…  L: 20
g: 9.8…
Enter value or press SOLVE
EDIT VARS INFO SOLVE
```

If the equation is dimensionally consistent, you can add units to the input values, as shown in the figure below. However, you must add those units to the initial guess in the solution. Thus, in the example below we place 0_m in the D: field before solving the problem. The solution is shown in the screen to the right:



```
SOLVE EQUATION
Eq: hf=f*V^2*L/(2*g*D)
hf: 2_m  ε: .00…  D: 0_m
Q: .5_…  Nu: .0…  L: 10_m
g: 9.8…
Enter value or press SOLVE
EDIT VARS INFO SOLVE
```



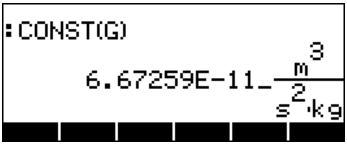
```
SOLVE EQUATION
Eq: hf=f*V^2*L/(2*g*D)
hf: 2_m  ε: .00…  D: .25…
Q: .5_…  Nu: .0…  L: 10_m
g: 9.8…
'.259210489698_m'
CANCL OK
```

Press ENTER to return to normal calculator display. The solution for D will be listed in the stack.

**Example 4 – Universal gravitation**

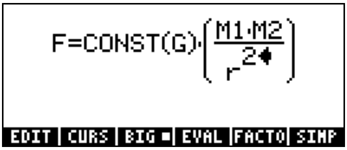
Newton's law of universal gravitation indicates that the magnitude of the attractive force between two bodies of masses $m_1$ and $m_2$ separated by a distance $r$ is given by the equation $F = G\cdot\dfrac{M_1 \cdot M_2}{r^2}$.

Here, G is the universal gravitational constant, whose value can be obtained through the use of the function CONST in the calculator by using:



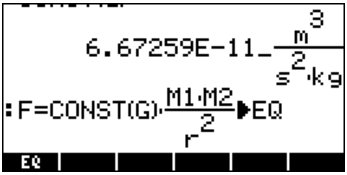
```
:CONST(G)
                 3
                m
6.67259E-11_─────
              2
             s ·kg
```

We can solve for any term in the equation (except G) by entering the equation as:



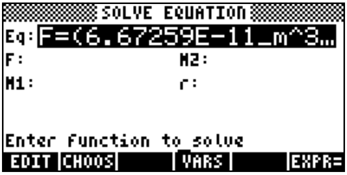
$$F=CONST(G)\cdot\left(\frac{M1\cdot M2}{r^2}\right)$$

This equation is then stored in EQ:



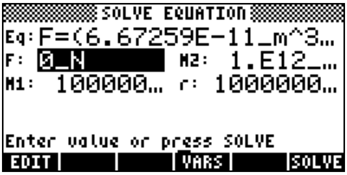
```
                 3
                m
6.67259E-11_─────
              2
             s ·kg
:F=CONST(G)·M1·M2 ▶EQ
            ─────
              2
             r
```

Launching the numerical solver for this equation results in an input form containing input fields for F, G, m1, m2, and r.



```
SOLVE EQUATION
Eq: F=(6.67259E-11_m^3…
F:            M2:
M1:           r:

Enter function to solve
```

Let's solve this problem using units with the following values for the known variables m1 = 1.0×$10^6$ kg, m2 = 1.0×$10^{12}$ kg, r = 1.0×$10^{11}$ m. Also, enter a value of 0_N in field F to ensure the proper solution using units in the calculator:



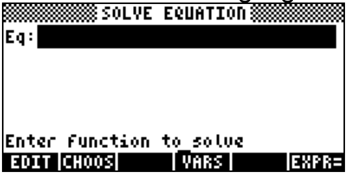
```
SOLVE EQUATION
Eq:F=(6.67259E-11_m^3…
F: 0_N          M2: 1.E12_…
M1: 100000…  r: 1000000…

Enter value or press SOLVE
```

Solve for F, and press to return to normal calculator display. The solution is F : 6.67259E-15_N, or F = 6.67259×10$^{-15}$ N.

**Note**: When using units in the numerical solver make sure that all the variables have the proper units, that the units are compatible, and that the equation is dimensionally homogeneous.

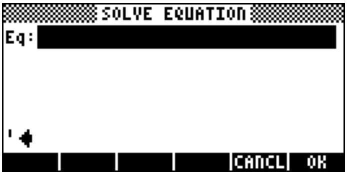
### Different ways to enter equations into EQ

In all the examples shown above we have entered the equation to be solved directly into variable EQ before activating the numerical solver. You can actually type the equation to be solved directly into the solver after activating it by editing the contents of the EQ field in the numerical solver input form. If variable EQ has not been defined previously, when you launch the numerical solver (↱ NUM.SLV OK), the EQ field will be highlighted:



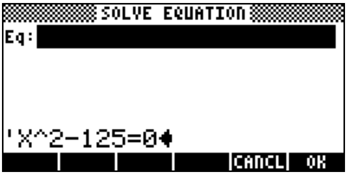
```
SOLVE EQUATION
Eq:

Enter function to solve
```

At this point you can either type a new equation by pressing EDIT. You will be provided with a set of apostrophes so that you can type the expression between them:

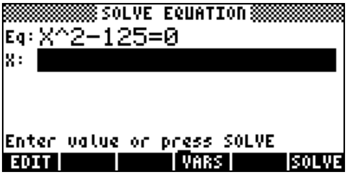


```
SOLVE EQUATION
Eq:

'◆
```

Type an equation, say X^2 - 125 = 0, directly on the stack, and press OK .



```
SOLVE EQUATION
Eq:

'X^2-125=0
                    CANCL  OK
```

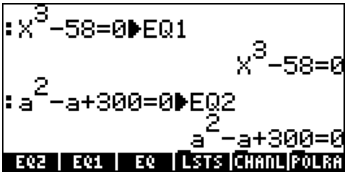


```
SOLVE EQUATION
Eq: X^2-125=0
X:

Enter value or press SOLVE
EDIT          VARS       SOLVE
```

At this point the equation is ready for solution.

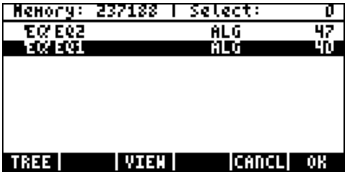
Alternatively, you can activate the equation writer after pressing EDIT to enter your equation. Press ENTER to return to the numerical solver screen.

Another way to enter an equation into the EQ variable is to select a variable already existing in your directory to be entered into EQ. This means that your equation would have to have been stored in a variable name previously to activating the numerical solver. For example, suppose that we have entered the following equations into variables EQ1 and EQ2:



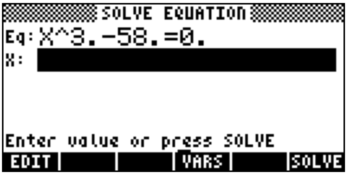
```
:X^3-58=0▶EQ1
                    X^3-58=0
:a^2-a+300=0▶EQ2
                 a^2-a+300=0
EQ2  EQ1  EQ  LSTS  CHANL  POLRA
```

Now, launch the numerical solver (↱ NUM.SLV OK, and highlight the EQ field. At this point press the CHOOS soft menu key. Use the up and down arrow keys (▲▼) to select, say, variable EQ1:



```
Memory: 237188 | Select:     0
 EQ EQ2            ALG      47
 EQ EQ1            ALG      40

TREE      VIEW      CANCL  OK
```

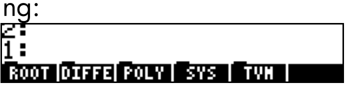
Press OK after selecting EQ1 to load into variable EQ in the solver. The new equation is ready to be solved.



```
SOLVE EQUATION
Eq: X^3.-58.=0.
X:

Enter value or press SOLVE
EDIT          VARS       SOLVE
```

# The SOLVE soft menu

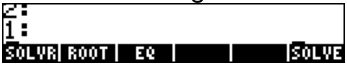
The SOLVE soft menu allows access to some of the numerical solver functions through the soft menu keys. To access this menu use in RPN mode: 74 MENU, or in ALG mode: MENU(74). Alternatively, you can use ↱(hold) 7 to activate the SOLVE soft menu. The sub-menus provided by the SOLVE soft menu are the following:



```
2:
1:
```

## The ROOT sub-menu

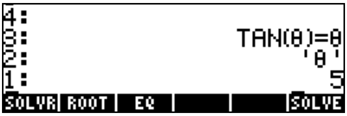
The ROOT sub-menu include the following functions and sub-menus:



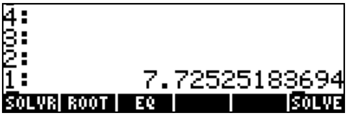
```
2:
1:
```

## Function ROOT

Function ROOT is used to solve an equation for a given variable with a starting guess value. In RPN mode the equation will be in stack level 3, while the variable name will be located in level 2, and the initial guess in level 1. The following figure shows the RPN stack before and after activating function ROOT:

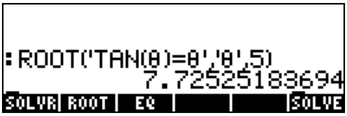


```
4:
3:                 TAN(θ)=θ
2:                      'θ'
1:                        5
```



```
4:
3:
2:
1:            7.72525183694
```

In ALG mode, you would use ROOT('TAN(θ)=θ','θ',5) to activate function ROOT:



```
:ROOT('TAN(θ)=θ','θ',5)
              7.72525183694
```

## Variable EQ

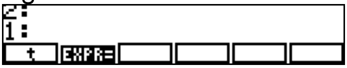
The soft menu key EQ in this sub-menu is used as a reference to the variable EQ. Pressing this soft menu key is equivalent to using function RCEQ (ReCall EQ).

## The SOLVR sub-menu

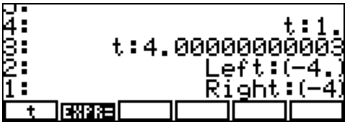
The SOLVR sub-menu activates the soft-menu solver for the equation currently stored in EQ.  Some examples are shown next:

Example 1 - Solving the equation $t^2$-5t = -4

For example, if you store the equation ‘t^2-5*t=-4’ into EQ, and press SOLVR, it will activate the following menu:



This result indicates that you can solve for a value of t for the equation listed at the top of the display.  If you try, for example, ↰[  t  ], it will give you the result t: 1., after briefly flashing the message “Solving for t.”  There is a second root to this equation, which can be found by changing the value of t, before solving for it again.  Do the following: 10 [  t  ], then press ↰[  t  ]. The result is now, t: 4.0000000003.  To verify this result, press the soft menu key labeled EXPR=, which evaluates the expression in EQ for the current value of t.  The results in this case are:

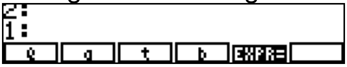


```
4:                  t:1.
3:       t:4.0000000003
2:            Left:(-4.)
1:            Right:(-4)
```

To exit the SOLVR environment, press VAR.  The access to the SOLVE menu is lost at this point, so you have to activate it once more as indicated earlier, to continue with the exercises below.

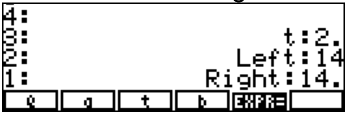
Example 2 - Solving the equation Q = $at^2$+bt

It is possible to store in EQ, an equation involving more than one variable, say, ‘Q = at^2 + bt’.   In this case, after activating the SOLVE soft menu, and pressing ROOT SOLVR, you will get the following screen:



Within this SOLVR environment you can provide values for any of the variables listed by entering the value in the stack and pressing the corresponding soft-menu keys.  For example, say you enter the values Q = 14, a = 2, and b = 3.  You would use:  14 [   Q   ], 2 [   a   ], 3 [   b   ].

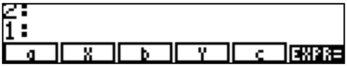
As variables Q, a, and b, get assigned numerical values, the assignments are listed in the upper left corner of the display. At this point we can solve for t, by using ↰[ t ]. The result is t: 2. Pressing EXPR= shows the results:



```
4:
3:            t:2.
2:         Left:14
1:       Right:14.
```

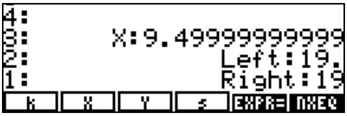
Example 3 - Solving two simultaneous equations, one at a time

You can also solve more than one equation by solving one equation at a time, and repeating the process until a solution is found. For example, if you enter the following list of equations into variable EQ: { 'a*X+b*Y = c', 'k*X*Y=s'}, the keystroke sequence ROOT SOLVR, within the SOLVE soft menu, will produce the following screen:



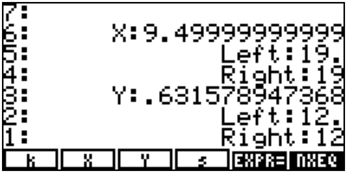
```
2:
1:
```

The first equation, namely, a*X + b*Y = c, will be listed in the top part of the display. You can enter values for the variables a, b, and c, say: 2 [ a ] 5 [ b ] 19 [ c ]. Also, since we can only solve one equation at a time, let's enter a guess value for Y, say, 0 [ Y ], and solve for X, by using ↰[ X ]. This gives the value, X: 9.4999…. To check the value of the equation at this point, press EXPR=. The results are: Left: 19, Right: 19. To solve the next equation, press NXT NXEQ. The screen shows the soft menu keys as:



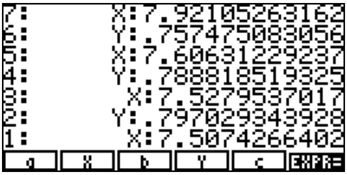
```
4:
3:      X:9.49999999999
2:            Left:19.
1:           Right:19
```

Say we enter the values k = 2, s = 12. Then solve for Y, and press EXPR=. The results are now, Y:

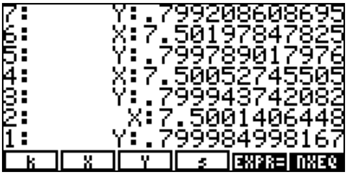


```
7:
6:      X:9.49999999999
5:            Left:19.
4:           Right:19
3:      Y:.631578947368
2:            Left:12.
1:           Right:12
```

We then continue moving from the first to the second equation, back and forth, solving the first equation for X and the second for Y, until the values of X and Y converge to a solution. To move from equation to equation use NXEQ. To solve for X and Y use ↰[ X ], and ↰[ Y ], respectively. The following sequence of solutions is produced:



```
7:        X:7.92105263162
6:        Y:.757475083056
5:        X:7.60631229237
4:        Y:.788818519325
3:         X:7.5279537017
2:        Y:.797029343928
1:         X:7.5074266402
```



```
7:        Y:.799208608695
6:        X:7.50197847825
5:        Y:.799789017976
4:        X:7.50052745505
3:        Y:.799943742082
2:         X:7.5001406448
1:        Y:.799984998167
```

After solving the two equations, one at a time, we notice that, up to the third decimal, X is converging to a value of 7.500, while Y is converging to a value o 0.799.

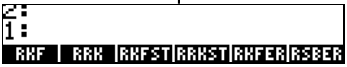
### Using units with the SOLVR sub-menu

These are some rules on the use of units with the SOLVR sub-menu:

- Entering a guess with units for a given variable, will introduce the use of those units in the solution.
- If a new guess is given without units, the units previously saved for that particular variable are used.
- To remove units enter a number without units in a list as the new guess, i.e., use the format { number }.
- A list of numbers can be given as a guess for a variable. In this case, the units takes the units used belong to the last number in the list. For example, entering { 1.41_ft 1_cm 1_m } indicates that meters (m) will be used for that variable.
- The expression used in the solution must have consistent units, or an error will result when trying to solve for a value.

## The DIFFE sub-menu

The DIFFE sub-menu provides a number of functions for the numerical solution of differential equations. The functions provided are the following:

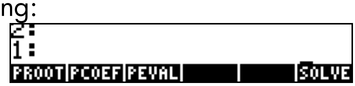


```
2:
1:
```

These functions are presented in detail in Chapter 16.

## The POLY sub-menu

The POLY sub-menu performs operations on polynomials. The functions included are the following:



```
2:
1:
```

### Function PROOT

This function is used to find the roots of a polynomial given a vector containing the polynomial coefficients in decreasing order of the powers of the independent variable. In other words, if the polynomial is $a_nx^n + a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x + a_0$, the vector of coefficients should be entered as $[a_n, a_{n-1}, \dots , a_2, a_1 , a_0]$. For example, the roots of the polynomial whose coefficients are [1, -5, 6] are [2, 3].

### Function PCOEF

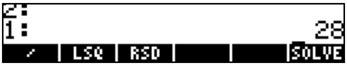
This function produces the coefficients $[a_n, a_{n-1}, \dots , a_2, a_1 , a_0]$ of a polynomial $a_nx^n + a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x + a_0$, given a vector of its roots $[r_1, r_2, \dots, r_n]$. For example, a vector whose roots are given by [-1, 2, 2, 1, 0], will produce the following coefficients: [1, -4, 3, 4, -4, 0]. The polynomial is $x^5 - 4x^4 + 3x^3 + 4x^2 - 4x$.

### Function PEVAL

This function evaluates a polynomial, given a vector of its coefficients, $[a_n, a_{n-1}, \dots , a_2, a_1 , a_0]$, and a value $x_0$, i.e., PEVAL calculates $a_nx_0^n + a_{n-1}x_0^{n-1} + \dots + a_2x_0^2 + a_1x_0 + a_0$. For example, for coefficients [2, 3, -1, 2] and a value of 2, PEVAL returns the value 28.

## The SYS sub-menu

The SYS sub-menu contains a listing of functions used to solve linear systems. The functions listed in this sub-menu are:

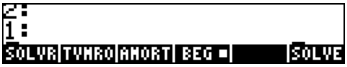


```
2:
1:                 28
```

These functions are presented in detail in Chapter 11.

## The TVM sub-menu

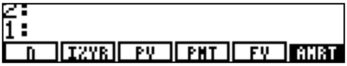
The TVM sub-menu contains functions for calculating Time Value of Money. This is an alternative way to solve FINANCE problems (see Chapter 6). The functions available are shown next:



```
2:
1:
```

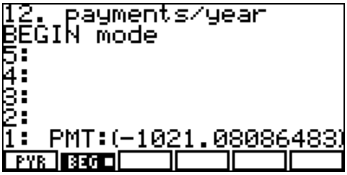
### The SOLVR sub-menu

The SOLVR sub-menu in the TVM sub-menu will launch the solver for solving TVM problems. For example, pressing SOLVR, at this point, will trigger the following screen:



```
2:
1:
```

As an exercise, try using the values n = 10, I%YR = 5.6, PV = 10000, and FV = 0, and enter ← [ PMT ] to find PMT = -1021.08…. Pressing NXT, produces the following screen:



```
12. payments/year
BEGIN mode
5:
4:
3:
2:
1:  PMT:(-1021.08086483)
```

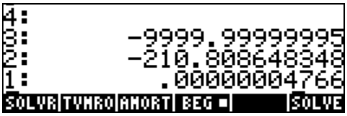
Press VAR to exit the SOLVR environment. Find your way back to the TVM sub-menu within the SOLVE sub-menu to try the other functions available.

### Function TVMROOT

This function requires as argument the name of one of the variables in the TVM problem. The function returns the solution for that variable, given that the other variables exist and have values stored previously. For example, having solved a TVM problem above, we can solve for, say, 'N', as follows: [ ' ] ALPHA N ENTER TVMRO. The result is 10.

### Function AMORT

This function takes a value representing a period of payment (between 0 and n) and returns the principal, interest, and balance for the values currently stored in the TVM variables. For example, with the data used earlier, if we activate function AMORT for a value of 10, we get:



```
4:
3:       -9999.99999995
2:       -210.808648348
1:        .00000004766
```

### Function BEG

If selected, the TMV calculations use payments at the beginning of each period.  If deselected, the TMV calculations use payments at the end of each period.

# Chapter 7
# Solving multiple equations

Many problems of science and engineering require the simultaneous solutions of more than one equation. The calculator provides several procedures for solving multiple equations as presented below. Please notice that no discussion of solving systems of linear equations is presented in this chapter. Linear systems solutions will be discussed in detail in subsequent chapters on matrices and linear algebra.

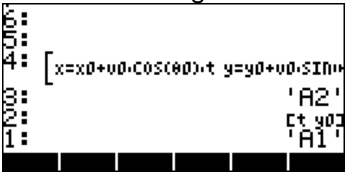
## Rational equation systems

Equations that can be re-written as polynomials or rational algebraic expressions can be solved directly by the calculator by using the function SOLVE. You need to provide the list of equations as elements of a vector. The list of variables to solve for must also be provided as a vector. Make sure that the CAS is set to mode Exact before attempting a solution using this procedure. Also, the more complicated the expressions, the longer the CAS takes in solving a particular system of equations. Examples of this application follow:

### Example 1 – Projectile motion

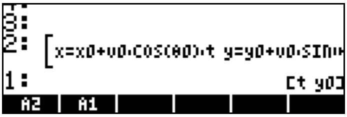
Use function SOLVE with the following vector arguments, the first being the list of equations: ['x = x0 + v0*COS(θ0)*t' 'y =y0+v0*SIN(θ0)*t – g*t^2/2'] ENTER , and the second being the variables to solve for, say t and y0, i.e., ['t' 'y0'].

The solution in this case will be provided using the RPN mode. The only reason being that we can build the solution step by step. The solution in the ALG mode is very similar. First, we store the first vector (equations) into variable A2, and the vector of variables into variable A1. The following screen shows the RPN stack before saving the variables.



```
6:
5:
4: [x=x0+v0·COS(θ0)·t y=y0+v0·SIN…
3:                              'A2'
2:                            [t y0]
1:                              'A1'
```

At this point, we need only press STO▶ twice to store these variables.
To solve, first change CAS mode to Exact, then, list the contents of A2 and A1, in that order: A2 A1 .

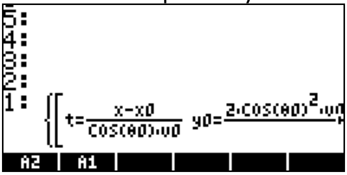


```
3:
2: [x=x0+v0·COS(θ0)·t y=y0+v0·SIN▸
1:                          [t y0]
 A2 │ A1 │    │    │    │
```

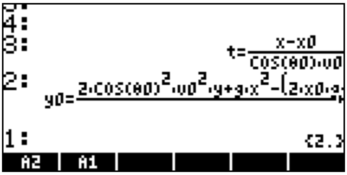
Use command SOLVE at this point (from the S.SLV menu: ← S.SLV ) After about 40 seconds, maybe more, you get as result a list:

{ 't = (x-x0)/(COS(θ0)*v0)'
'y0 = (2*COS(θ0)^2*v0^2*y+(g*x^2(2*x0*g+2*SIN(   0))*COS(θ0)*v0^2)*x+
(x0^2*g+2*SIN(θ0)*COS(θ0)*v0^2*x0)))/(2*COS(θ0)^2*v0^2)']}

Press EVAL to remove the vector from the list, then use command OBJ→, to get the equations listed separately in the stack.



```
5:
4:
3:
2:
1:  {[t=    x-x0      y0= 2·COS(θ0)^2·v0
        ─────────         ─────────────▸
        COS(θ0)·v0
 A2 │ A1 │    │    │    │
```



```
4:
3:                        t=    x-x0
                            ──────────
                            COS(θ0)·v0
2:
  y0= 2·COS(θ0)^2·v0^2·y+g·x^2-(2·x0·g▸
      ─────────────────────────────────▸
1:                                  {2.▸
 A2 │ A1 │    │    │    │
```

**Note**: This method worked fine in this example because the unknowns t and y0 were algebraic terms in the equations. This method would not work for solving for θ0, since θ0 belongs to a transcendental term.

## Example 2 – Stresses in a thick wall cylinder

Consider a thick-wall cylinder for inner and outer radius *a* and *b*, respectively, subject to an inner pressure $P_i$ and outer pressure $P_o$. At any radial distance *r* from the cylinder's axis the normal stresses in the radial and transverse directions, $\sigma_{rr}$ and $\sigma_{\theta\theta}$, respectively, are given by

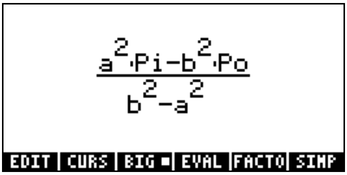
$$\sigma_{\theta\theta} = \frac{a^2 \cdot P_i - b^2 \cdot P_o}{b^2 - a^2} + \frac{a^2 \cdot b^2 \cdot (P_i - P_o)}{r^2 \cdot (b^2 - a^2)},$$

$$\sigma_{rr} = \frac{a^2 \cdot P_i - b^2 \cdot P_o}{b^2 - a^2} - \frac{a^2 \cdot b^2 \cdot (P_i - P_o)}{r^2 \cdot (b^2 - a^2)}.$$

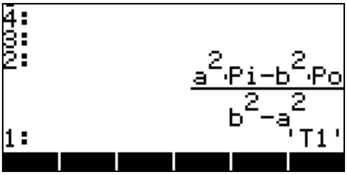
Notice that the right-hand sides of the two equations differ only in the sign between the two terms. Therefore, to write these equations in the calculator, I suggest you type the first term and store in a variable T1, then the second term, and store it in T2. Writing the equations afterwards will be matter of recalling the contents of T1 and T2 to the stack and adding and subtracting them.

Here is how to do it with the equation writer:

Enter and store term T1:

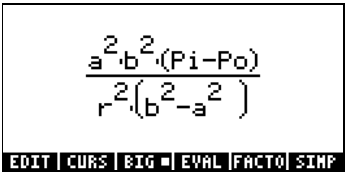


$$\frac{a^2\cdot Pi-b^2\cdot Po}{b^2-a^2}$$

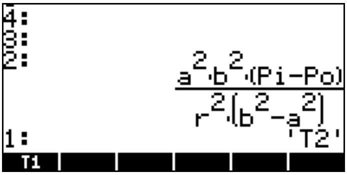


4:
3:
2: $\frac{a^2\cdot Pi-b^2\cdot Po}{b^2-a^2}$
1: 'T1'

Enter and store term T2:



$$\frac{a^2\cdot b^2\cdot (Pi-Po)}{r^2\cdot (b^2-a^2\ )}$$

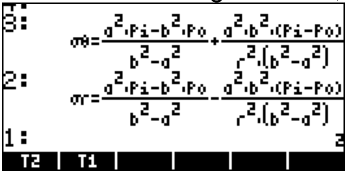


4:
3:
2: $\frac{a^2\cdot b^2\cdot (Pi-Po)}{r^2\cdot (b^2-a^2)}$
1: 'T2'

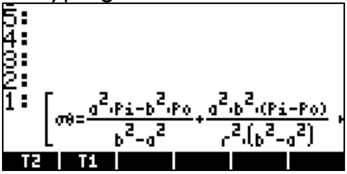
Notice that we are using the RPN mode in this example, however, the procedure in the ALG mode should be very similar. Create the equation for $\sigma_{\theta\theta}$: VAR T1 T2 + ALPHA ↱ S ALPHA ↱ T ENTER ▶ ↱ =

Create the equation for $\sigma_{rr}$: VAR T1 T2 − ALPHA ↱ S ALPHA ↰ R ENTER ▶ ↱ =

Put together a vector with the two equations, using function →ARRY (find it using the command catalog ↱ CAT ) after typing a 2 :

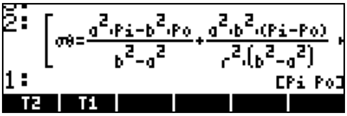


3: $\sigma\theta=\frac{a^2\cdot Pi-b^2\cdot Po}{b^2-a^2}+\frac{a^2\cdot b^2\cdot (Pi-Po)}{r^2\cdot (b^2-a^2)}$
2: $\sigma r=\frac{a^2\cdot Pi-b^2\cdot Po}{b^2-a^2}-\frac{a^2\cdot b^2\cdot (Pi-Po)}{r^2\cdot (b^2-a^2)}$
1: 2



5:
4:
3:
2:
1: $\left[\sigma\theta=\frac{a^2\cdot Pi-b^2\cdot Po}{b^2-a^2}+\frac{a^2\cdot b^2\cdot (Pi-Po)}{r^2\cdot (b^2-a^2)}\right.$ ▸

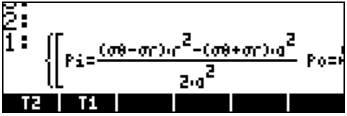
Now, suppose that we want to solve for $P_i$ and $P_o$, given $a$, $b$, r, $\sigma_{rr}$, and $\sigma_{\theta\theta}$. We enter a vector with the unknowns:



2: $\left[\sigma\theta=\frac{a^2\cdot Pi-b^2\cdot Po}{b^2-a^2}+\frac{a^2\cdot b^2\cdot (Pi-Po)}{r^2\cdot (b^2-a^2)}\right.$ ▸
1: [Pi Po]

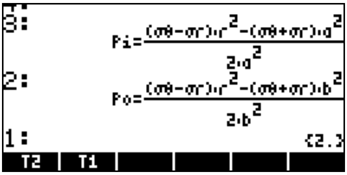
To solve for $P_i$ and $P_o$, use the command SOLVE from the S.SLV menu (← S.SLV ), it may take the calculator a minute to produce the result:

{['Pi=-(((σθ-σr)*r^2-(σθ+σr)*a^2)/(2*a^2))'
'Po=-(((σθ-σr)*r^2-(σθ+σr)*b^2)/(2*b^2))' ] }, i.e.,



```
2:
1: {[Pi=(σθ-σr)·r²-(σθ+σr)·a² / 2·a²   Po=
 T2 | T1 |
```

Notice that the result includes a vector [ ] contained within a list { }. To remove the list symbol, use EVAL. Finally, to decompose the vector, use function OBJ→. The result is:

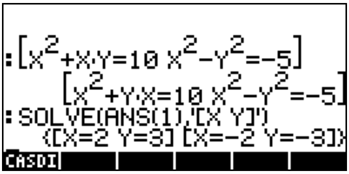


```
3:        Pi=(σθ-σr)·r²-(σθ+σr)·a² / 2·a²
2:        Po=(σθ-σr)·r²-(σθ+σr)·b² / 2·b²
1:                                  {2.}
 T2 | T1 |
```

These two examples constitute systems of linear equations that can be handled equally well with function LINSOLVE (see Chapter 11). The following example shows function SOLVE applied to a system of polynomial equations.

### Example 3 - System of polynomial equations

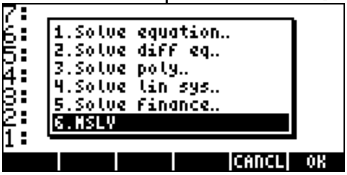
The following screen shot shows the solution of the system $X^2+XY=10$, $X^2-Y^2=-5$, using function SOLVE:



```
:[X²+X·Y=10 X²-Y²=-5]
     [X²+Y·X=10 X²-Y²=-5]
:SOLVE(ANS(1),'[X Y]')
   {[X=2 Y=3] [X=-2 Y=-3]}
CASDI|
```

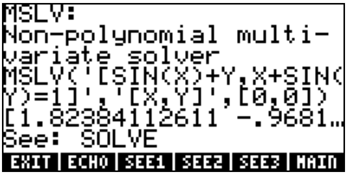
## Solution to simultaneous equations with MSLV

Function MSLV is available as the last option in the → NUM.SLV menu:



```
7:
6:  1.Solve equation..
5:  2.Solve diff eq..
4:  3.Solve poly..
3:  4.Solve lin sys..
2:  5.Solve finance..
    6.MSLV
1:
                        |CANCL| OK
```

The help-facility entry for function MSLV is shown next:



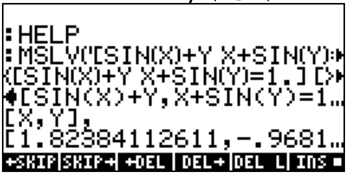
```
MSLV:
Non-polynomial multi-
variate solver
MSLV('[SIN(X)+Y,X+SIN(
Y)=1]','[X,Y]',[0,0])
[1.82384112611 -.9681…
See: SOLVE
```

## Example 1 – Example from the help facility

As with all function entries in the help facility, there is an example attached to the MSLV entry as shown above. Notice that function MSLV requires three arguments:

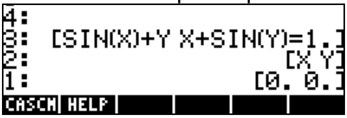
1. A vector containing the equations, i.e., '[SIN(X)+Y,X+SIN(Y)=1]'
2. A vector containing the variables to solve for, i.e., '[X,Y]'
3. A vector containing initial values for the solution, i.e., the initial values of both X and Y are zero for this example.

In ALG mode, press ECHO to copy the example to the stack, press ENTER to run the example. To see all the elements in the solution you need to activate the line editor by pressing the down arrow key (▼):



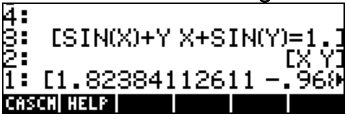
```
:HELP
:MSLV('[SIN(X)+Y X+SIN(Y):▸
{[SIN(X)+Y X+SIN(Y)=1.] [>▸
◆[SIN(X)+Y,X+SIN(Y)=1…
[X,Y],
[1.82384112611,-.9681…
```

In RPN mode, the solution for this example is produced by using:



```
4:
3:  [SIN(X)+Y X+SIN(Y)=1.]
2:                    [X Y]
1:                 [0. 0.]
```

Activating function MSLV results in the following screen.



```
4:
3:  [SIN(X)+Y X+SIN(Y)=1.]
2:                    [X Y]
1: [1.82384112611 -.96▸
```

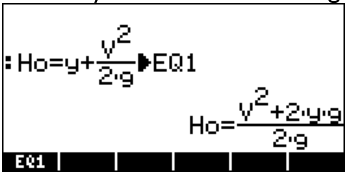
You may have noticed that, while producing the solution, the screen shows intermediate information on the upper left corner. Since the solution provided by MSLV is numerical, the information in the upper left corner shows the results of the iterative process used to obtain a solution. The final solution is *X = 1.8238, Y = -0.9681.*

## Example 2 - Entrance from a lake into an open channel

This particular problem in open channel flow requires the simultaneous solution of two equations, the equation of energy: $H_o = y + \frac{V^2}{2g}$, and Manning's equation: $Q = \frac{Cu}{n} \cdot \frac{A^{5/3}}{P^{2/3}} \cdot \sqrt{S_o}$. In these equations, $H_o$ represents the energy head (m, or ft) available for a flow at the entrance to a channel, y is the flow depth (m or ft), V = Q/A is the flow velocity (m/s or ft/s), Q is the volumetric discharge ($m^3$/s or $ft^3$/s), A is the cross-sectional area ($m^2$ or $ft^2$), $C_u$ is a coefficient that depends on the system of units ($C_u$ = 1.0 for the SI, $C_u$ = 1.486 for the English system of units), n is the Manning's coefficient, a measure of the channel surface roughness (e.g., for concrete, n = 0.012), P is the wetted perimeter of the cross section (m or ft), $S_o$ is the slope of the channel bed expressed as a decimal fraction. For a trapezoidal channel, as shown below, the area is given by $A = (b + my)y$, while the wetted perimeter is given by $P = b + 2y\sqrt{1+m^2}$, where b is the bottom width (m or ft), and m is the side slope (1V:mH) of the cross section.

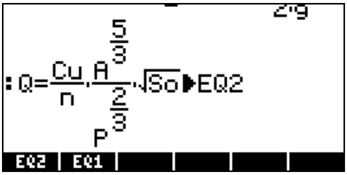
Typically, one has to solve the equations of energy and Manning's simultaneously for y and Q. Once these equations are written in terms of the primitive variables b, m, y, g, $S_o$, n, Cu, Q, and $H_o$, we are left with a system of equations of the form $f_1(y,Q) = 0$, $f_2(y,Q) = 0$. We can build these two equations as follows.

We assume that we will be using the ALG mode in the calculator, although defining the equations and solving them with MSLV is very similar in the RPN mode. Create a sub-directory, say CHANL (for open CHANneL), and within that sub-directory define the following variables:



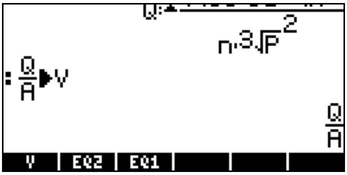
:$Ho=y+\frac{V^2}{2\cdot g}$▶EQ1

$Ho=\frac{V^2+2\cdot y\cdot g}{2\cdot g}$



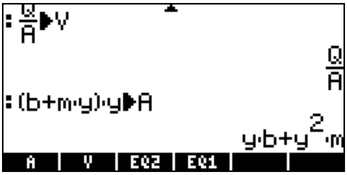
$2\cdot g$

:$Q=\frac{Cu}{n}\cdot\frac{A^{\frac{5}{3}}}{P^{\frac{2}{3}}}\cdot\sqrt{So}$▶EQ2



$:\frac{Q}{A}▶V$

$\frac{Q}{A}$

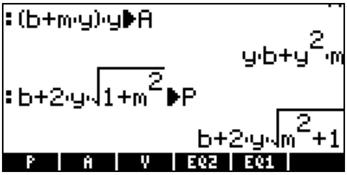


$:\frac{Q}{A}▶V$

$\frac{Q}{A}$

$:(b+m\cdot y)\cdot y▶A$

$y\cdot b+y^2\cdot m$



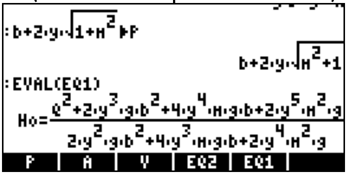
$:(b+m\cdot y)\cdot y▶A$

$y\cdot b+y^2\cdot m$

$:b+2\cdot y\cdot\sqrt{1+m^2}▶P$

$b+2\cdot y\cdot\sqrt{m^2+1}$

To see the original equations, EQ1 and EQ2, in terms of the primitive variables listed above, we can use function EVAL applied to each of the equations, i.e., EVAL EQ1 EVAL EQ2. The equations are listed in the stack as follows (small font option selected):

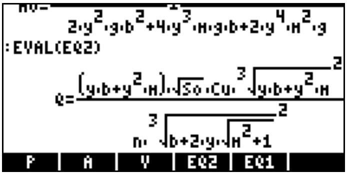


$:b+2\cdot y\cdot\sqrt{1+m^2}▶P$

$b+2\cdot y\cdot\sqrt{m^2+1}$

:EVAL(EQ1)

$Ho=\frac{Q^2+2\cdot y^3\cdot g\cdot b^2+4\cdot y^4\cdot m\cdot g\cdot b+2\cdot y^5\cdot m^2\cdot g}{2\cdot y^2\cdot g\cdot b^2+4\cdot y^3\cdot m\cdot g\cdot b+2\cdot y^4\cdot m^2\cdot g}$



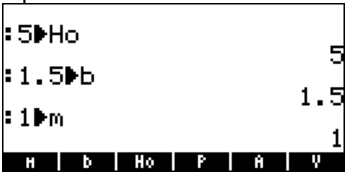
$2\cdot y^2\cdot g\cdot b^2+4\cdot y^3\cdot m\cdot g\cdot b+2\cdot y^4\cdot m^2\cdot g$

:EVAL(EQ2)

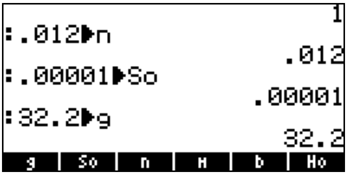
$Q=\frac{(y\cdot b+y^2\cdot m)\cdot\sqrt{So}\cdot Cu\cdot\sqrt[3]{(y\cdot b+y^2\cdot m)^2}}{n\cdot\sqrt[3]{(b+2\cdot y\cdot\sqrt{m^2+1})^2}}$

We can see that these equations are indeed given in terms of the primitive variables b, m, y, g, $S_o$, n, Cu, Q, and $H_o$.

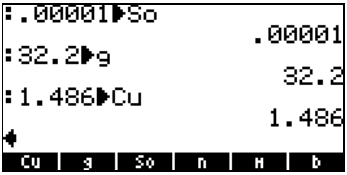
In order to solve for y and Q we need to give values to the other variables. Suppose we use $H_0$ = 5 ft, b = 1.5 ft, m = 1, n = 0.012, $S_0$ = 0.00001, g = 32.2, and Cu = 1.486. Before being able to use MSLV for the solution, we need to enter these values into the corresponding variable names. This can be accomplished as follows:



```
:5▶Ho
                      5
:1.5▶b
                    1.5
:1▶m
                      1
```

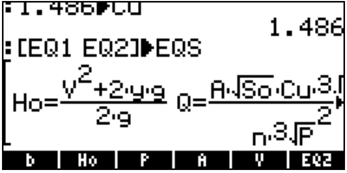


```
                      1
:.012▶n
                   .012
:.00001▶So
                 .00001
:32.2▶g
                   32.2
```



```
:.00001▶So
                .00001
:32.2▶g
                  32.2
:1.486▶Cu
                 1.486
```

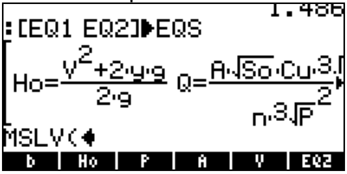
Now, we are ready to solve the equation. First, we need to put the two equations together into a vector. We can do this by actually storing the vector into a variable that we will call EQS (EQuationS):



```
:1.486▶Cu
                 1.486
:[EQ1 EQ2]▶EQS
```

$$\left[\ Ho=\frac{V^2+2\cdot y\cdot g}{2\cdot g}\quad Q=\frac{A\cdot\sqrt{So}\cdot Cu\cdot 3.}{n\cdot\sqrt[3]{P}^{2}}\right.$$

As initial values for the variables y and Q we will use y = 5 (equal to the value of $H_o$, which is the maximum value that y can take) and Q = 10 (this is a guess). To obtain the solution we select function MSLV from the NUM.SLV menu, e.g., → NUM.SLV 6 OK, to place the command in the screen:



```
                 1.486
:[EQ1 EQ2]▶EQS
```

$$\left[\ Ho=\frac{V^2+2\cdot y\cdot g}{2\cdot g}\quad Q=\frac{A\cdot\sqrt{So}\cdot Cu\cdot 3.}{n\cdot\sqrt[3]{P}^{2}}\right.$$

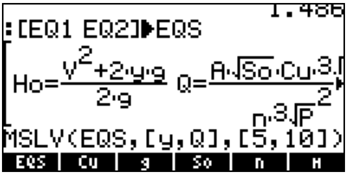
```
MSLV(◆
```

Next, we'll enter variable EQS: NXT NXT EQS, followed by vector [y,Q]:

→ , ← [] ALPHA ← Y → , ALPHA Q ▶

and by the initial guesses → , ← [] 5 → , 1 0.

Before pressing ENTER, the screen will look like this:

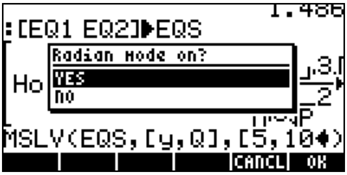


```
                 1.486
:[EQ1 EQ2]▶EQS
```

$$\left[\ Ho=\frac{V^2+2\cdot y\cdot g}{2\cdot g}\quad Q=\frac{A\cdot\sqrt{So}\cdot Cu\cdot 3.}{n\cdot\sqrt[3]{P}^{2}}\right.$$

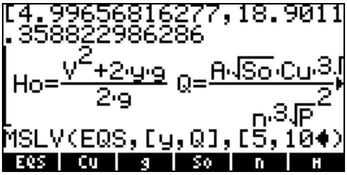
```
MSLV(EQS,[y,Q],[5,10])
```

Press ENTER to solve the system of equations. You may, if your angular measure is not set to radians, get the following request:



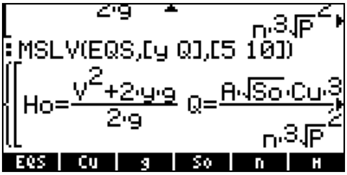
```
1.486
:[EQ1 EQ2]▶EQS
Radian mode on?
YES
no
MSLV(EQS,[y,Q],[5,10◆)
```

Press OK and allow the solution to proceed. An intermediate solution step may look like this:



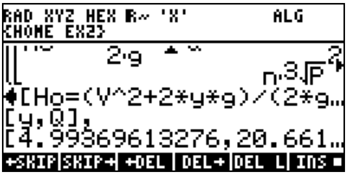
```
[4.99656816277,18.9011
.358822986286
Ho=(V^2+2·y·g)/(2·g)  Q=A·√So·Cu·3…/(n·3√P…)
MSLV(EQS,[y,Q],[5,10◆)
```

The vector at the top representing the current value of [y,Q] as the solution progresses, and the value .358822986286 representing the criteria for convergence of the numerical method used in the solution. If the system is well posed, this value will diminish until reaching a value close to zero. At that point a numerical solution would have been found. The screen, after MSLV finds a solution will look like this:



```
:MSLV(EQS,[y Q],[5 10])
Ho=(V^2+2·y·g)/(2·g)  Q=A·√So·Cu·3…/(n·3√P…)
```

The result is a list of three vectors. The first vector in the list will be the equations solved. The second vector is the list of unknowns. The third vector represents the solution. To be able to see these vectors, press the down-arrow key ▼ to activate the line editor. The solution will be shown as follows:



```
RAD XYZ HEX R∼ 'X'        ALG
{HOME EX2}
◆[Ho=(V^2+2*y*g)/(2*g…
[y,Q],
[4.99369613276,20.661…
```

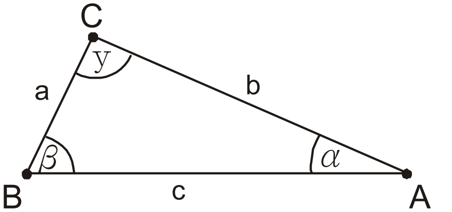
The solution suggested is [4.9936.., 20.661…]. This means, y = 4.99 ft, and Q = 20.661… ft$^3$/s. You can use the arrow keys (◀ ▶ ▲ ▼) to see the solution in detail.

# Using the Multiple Equation Solver (MES)

The multiple equation solver is an environment where you can solve a system of multiple equations by solving for one unknown from one equation at a time. It is not really a solver to simultaneous solutions, rather, it is a one-by-one solver of a number of related equations. To illustrate the use of the MES for solving multiple equations we present an application related to trigonometry in the next section. The examples shown here are developed in the RPN mode.

## Application 1 - Solution of triangles

In this section we use one important application of trigonometric functions: calculating the dimensions of a triangle. The solution is implemented in the calculator using the Multiple Equation Solver, or MES.
Consider the triangle ABC shown in the figure below.



The sum of the interior angles of any triangle is always 180°, i.e., $\alpha + \beta + \gamma =$ 180°. The sine law indicates that:

$$\frac{\sin\alpha}{a} = \frac{\sin\beta}{b} = \frac{\sin\gamma}{c}.$$

The cosine law indicates that:

$$a^2 = b^2 + c^2 - 2\cdot b\cdot c\cdot \cos\alpha,$$
$$b^2 = a^2 + c^2 - 2\cdot a\cdot c\cdot \cos\beta,$$
$$c^2 = a^2 + b^2 - 2\cdot a\cdot b\cdot \cos\gamma.$$

In order to solve any triangle, you need to know at least three of the following six variables: *a, b, c, $\alpha$, $\beta$, $\gamma$.* Then, you can use the equations of the sine law, cosine law, and sum of interior angles of a triangle, to solve for the other three variables.

If the three sides are known, the area of the triangle can be calculated with Heron's formula $A = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$ ,where *s* is known as the semi-perimeter of the triangle, i.e., $s = \frac{a+b+c}{2}$.

### Triangle solution using the Multiple Equation Solver (MES)

The Multiple Equation Solver (MES) is a feature that can be used to solve two or more coupled equations. It must be pointed out, however, that the MES does not solve the equations simultaneously. Rather, it takes the known variables, and then searches in a list of equations until it finds one that can be solved for one of the unknown variables. Then, it searches for another equation that can be solved for the next unknowns, and so on, until all unknowns have been solved for.

#### *Creating a working directory*

We will use the MES to solve for triangles by creating a list of equations corresponding to the sine and cosine laws, the law of the sum of interior angles, and Heron's formula for the area. First, create a sub-directory within HOME that we will call TRIANG, and move into that directory. See Chapter 2 for instructions on how to create a new sub-directory.

#### *Entering the list of equations*

Within TRIANG, enter the following list of equations either by typing them directly on the stack or by using the equation writer. (Recall that ALPHA ↱ A produces the character α, and ALPHA ↱ B produces the character β. The character γ needs to be ECHOed from ↱ CHARS ):

'SIN(α)/a = SIN(β)/b'
'SIN(α)/a = SIN(γ)/c'
'SIN(β)/b = SIN(γ)/c'
'c^2 = a^2+b^2-2*a*b*COS(γ)'
'b^2 = a^2+c^2-2*a*c*COS(β)'

'a^2 = b^2+c^2-2*b*c*COS(α)'
'α+β+γ = 180'
's = (a+b+c)/2'
'A = √ (s*(s-a)*(s-b)*(s-c))'

Then, enter the number 9, and create a list of equations by using: function →LIST (use the command catalog ↱ CAT ). Store this list in the variable EQ.

The variable EQ contains the list of equations that will be scanned by the MES when trying to solve for the unknowns.

### *Entering a window title*

Next, we will create a string variable to be called TITLE to contain the string "Triangle Solution", as follows:

| | |
|---|---|
| ↱ "" | Open double quotes in stack |
| ALPHA ALPHA ↰ ALPHA | Locks keyboard into lower-case alpha. |
| ↰ T R I A N G L E SPC | Enter text: Triangle_ |
| ↰ S O L U T I O N | Enter text: Solution |
| ENTER | Enter string "Triangle Solution" in stack |
| ' | Open single quotes in stack |
| ALPHA ALPHA T I T L E ENTER | Enter variable name 'TITLE' |
| STO▶ | Store string into 'TITLE' |

### *Creating a list of variables*

Next, create a list of variable names in the stack that will look like this:

`{ a b c α β γ A s }`

and store it in variable LVARI (List of VARIables). The list of variables represents the order in which the variables will be listed when the MES gets started. It must include all the variables in the equations, or it will not work with function MITM (see below). Here is the sequence of keystrokes to use to prepare and store this list:

Press VAR, if needed, to get your variables menu. Your menu should show the variables LVARI TITLE EQ .

### *Preparing to run the MES*

The next step is to activate the MES and try one sample solution. Before we do that, however, we want to set the angular units to DEGrees, if they are not already set to that, by typing ALPHA ALPHA D E G ENTER.

Next, we want to keep in the stack the contents of TITLE and LVARI, by using:

TITLE LVARI

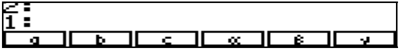
We will use the following MES functions

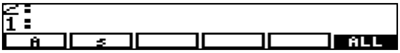
- MINIT: MES INITialization: initializes the variables in the equations stored in EQ.
- MITM: MES' Menu Item: Takes a title from stack level 2 and the list of variables from stack level 1 and places the title atop of the MES window, and the list of variables as soft menu keys in the order indicated by the list. In the present exercise, we already have a title ("Triangle Solution") and a list of variables ({ `a b c` α β γ `A s` }) in stack levels 2 and 1, respectively, ready to activate MITM.
- MSOLVR: MES SOLVER; activates the Multiple Equation Solver (MES) and waits for input by the user.

### *Running the MES interactively*

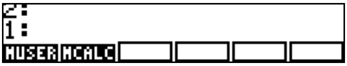
To get the MES started, with the variables TITLE and LVARI listed in the stack, activate command MINIT, then MITM, and finally, MSOLVR (find these functions in the catalog → CAT ).

The MES is launched with the following list of variables available (Press NXT to see the next list of variables):





Press NXT to see the third list of variables. You should see:



Press NXT once more to recover the first variable menu.

Let's try a simple solution of Case I, using a = 5, b = 3, c = 5. Use the following entries:

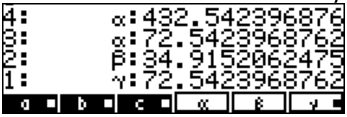
5 [ a ]     a:5 is listed in the top left corner of the display.
3 [ b ]     b:3 is listed in the top left corner of the display.
5 [ c ]     c:5 is listed in the top left corner of the display.
To solve for the angles use:
↰ [ α ]     Calculator reports Solving for α, and shows the result α: 72.5423968763.

**Note**: If you get a value that is larger than 180, try the following:

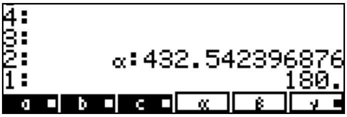
1 0 [ α ]     Re-initialize a to a smaller value.
↰ [ α ]     Calculator reports Solving for α

Next, we calculate the other two values:
↰ [ β ]     The result is β: 34.9152062474 .
↰ [ γ ]     The result is γ: 72.5423968763.

You should have the values of the three angles listed in stack levels 3 through 1. Press + twice to check that they add indeed to 180°.

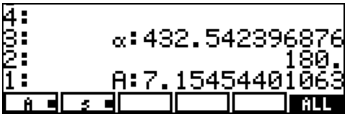


```
4:        α:432.542396876
3:        α:72.5423968762
2:        β:34.9152062475
1:        γ:72.5423968762
 a ▪ | b ▪ | c ▪ |  α  |  β  | γ ▪
```



```
4:
3:
2:        α:432.542396876
1:                   180.
 a ▪ | b ▪ | c ▪ |  α  |  β  | γ ▪
```

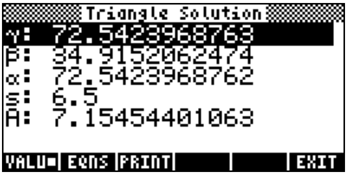
Press NXT to move to the next variables menu. To calculate the area use: ↰ [ A ]. The calculator first solves for all the other variables, and then finds the area as A: 7.15454401063.



```
4:
3:        α:432.542396876
2:                   180.
1:        A:7.15454401063
 A ▪ | s ▪ |     |     |     | ALL
```

**Note**: When a solution is found, the calculator reports the conditions for the solution as either Zero, or Sign Reversal. Other messages may occur if the calculator has difficulties finding a solution.

Pressing ↰ ALL will solve for all the variables, temporarily showing the intermediate results. Press ↱ ALL to see the solutions:

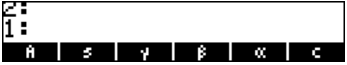


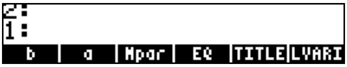
Triangle Solution
γ: 72.5423968763
β: 34.9152062474
α: 72.5423968762
s: 6.5
A: 7.15454401063

When done, press ON to return to the MES environment. Press VAR to exit the MES environment and return to the normal calculator display.

***Organizing the variables in the sub directory***

Your variable menu will now contain the variables (press NXT to see the second set of variables):





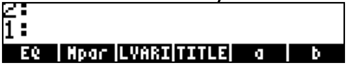
Variables corresponding to all the variables in the equations in EQ have been created. There is also a new variable called *Mpar* (MES parameters), which contains information regarding the setting up of the MES for this particular set of equations. If you use ↱ Mpar to see the contents of the variable *Mpar*. You will get the cryptic message: Library Data. The meaning of this is that the MES parameters are coded in a binary file, which cannot be accessed by the editor.

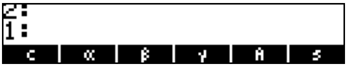
Next, we want to place them in the menu labels in a different order than the one listed above, by following these steps:

1. Create a list containing { EQ Mpar LVARI TITLE }, by using:
   ↰ { } EQ Mpar LVARI TITLE ENTER
2. Place contents of LVARI in the stack, by using: LVARI.
3. Join the two lists by pressing +.

Use function ORDER (use the command catalog ↱ CAT ) to order the variables as shown in the list in stack level 1.

4. Press VAR to recover your variables list. It should now look like this:





5. Press NXT to recover the first variable menu.

**Programming the MES triangle solution using User RPL**

To facilitate activating the MES for future solutions, we will create a program that will load the MES with a single keystroke. The program should look like this: << DEG MINIT TITLE LVARI MITM MSOLVR >>, and can be typed in by using:

| | |
|---|---|
| ↱ « » | Opens the program symbol |
| ALPHA ALPHA | Locks alphanumeric keyboard |
| D E G SPC | Type in DEG (angular units set to DEGrees) |
| M I N I T SPC | Type in MINIT_ |
| ALPHA | Unlocks alphanumeric keyboard |
| TITLE | List the name TITLE in the program |
| LVARI | List the name LVARI in the program |
| ALPHA ALPHA | Locks alphanumeric keyboard |
| M I T M SPC | Type in MITM_ |
| M S O L V R | Type in MSOLVR |
| ENTER | Enter program in stack |

Store the program in a variable called TRISOL, for TRIangle SOLution, by using: ' ALPHA ALPHA T R I S O L ENTER STO▶

Press VAR, if needed, to recover your list of variables. A soft key label TRISO should be available in your menu.

***Running the program – solution examples***

To run the program, press the TRISO soft menu key. You will now have the MES menu corresponding to the triangle solution. Let's try examples of the three cases listed earlier for triangle solution.

Example 1 – Right triangle

Use a = 3, b = 4, c = 5. Here is the solution sequence:

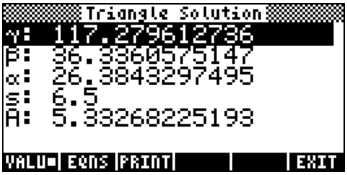
| | |
|---|---|
| 3 [ a ] 4 [ b ] 5 [ c ] | To enter data |
| ↰ [ α ] | The result is α: 36.8698976458 |
| ↰ [ β ] | The result is β: 53.1301023541. |
| ↰ [ γ ] | The result is γ: 90. |
| NXT | To move to the next variables menu. |
| [↰][ A ] | The result is A: 6. |
| NXT NXT | To move to the next variables menu. |

Example 2 - Any type of triangle

Use a = 3, b = 4, c = 6. The solution procedure used here consists of solving for all variables at once, and then recalling the solutions to the stack:

| | |
|---|---|
| VAR TRISO | To clear up data and re-start MES |
| 3 [ a ] 4 [ b ] 6 [ c ] | To enter data |
| NXT | To move to the next variables menu. |
| ↰ ALL | Solve for all the unknowns. |
| ↱ ALL | Show the solution: |

The solution is:

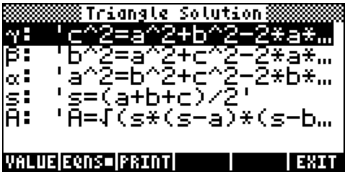


```
Triangle Solution
γ:  117.279612736
β:  36.3360575147
α:  26.3843297495
s:  6.5
A:  5.33268225193
VALU■ EQNS PRINT        EXIT
```

At the bottom of the screen, you will have the soft menu keys:

VALU■ EQNS PRINT EXIT

The square dot in VALU■ indicates that the values of the variables, rather than the equations from which they were solved, are shown in the display. To see the equations used in the solution of each variable, press the EQNS soft menu key. The display will now look like this:



```
Triangle Solution
γ:  'c^2=a^2+b^2-2*a*…
β:  'b^2=a^2+c^2-2*a*…
α:  'a^2=b^2+c^2-2*b*…
s:  's=(a+b+c)/2'
A:  'A=√(s*(s-a)*(s-b…
VALUE EQNS■ PRINT        EXIT
```

The soft menu key PRINT is used to print the screen in a printer, if available. And EXIT returns you to the MES environment for a new solution, if needed. To return to normal calculator display, press VAR.

The following table of triangle solutions shows the data input in bold face and the solution in italics. Try running the program with these inputs to verify the solutions. Please remember to press VAR TRISO at the end of each solution to clear up variables and start the MES solution again. Otherwise, you may carry over information from the previous solution that may wreck havoc with your current calculations.

| ***a*** | ***b*** | ***c*** | $\alpha(^o)$ | $\beta(^o)$ | $\gamma(^o)$ | ***A*** |
|---|---|---|---|---|---|---|
| **2.5** | *6.9837* | **7.2** | *20.299* | **75** | *84.771* | *8.6933* |
| **7.2** | **8.5** | *14.26* | *22.616* | **27** | *130.38* | *23.309* |
| *21.92* | **17.5** | **13.2** | **90** | *52.97* | **37.03** | *115.5* |
| *41.92* | **23** | *29.6* | **75** | **32** | *73* | *328.81* |
| *10.27* | *3.26* | **10.5** | *77* | **18** | **85** | *16.66* |
| **17** | **25** | **32** | *31.79* | *50.78* | *97.44* | *210.71* |

### Adding an INFO button to your directory

An information button can be useful for your directory to help you remember the operation of the functions in the directory. In this directory, all we need to remember is to press TRISO to get a triangle solution started. You may want to type in the following program: <<"Press [TRISO] to start." MSGBOX >>, and store it in a variable called INFO. As a result, the first variable in your directory will be the INFO button.

## Application 2 - Velocity and acceleration in polar coordinates

Two-dimensional particle motion in polar coordinates often involves determining the radial and transverse components of the velocity and acceleration of the particle given r, r' = dr/dt, r" = $d^2r/dt^2$, θ, θ' = d θ /dt, and, θ" = $d^2\theta/dt^2$. The following equations are used:

$$v_r = \dot{r} \qquad a_r = \ddot{r} - r\dot{\theta}^2$$

$$v_\theta = r\dot{\theta} \qquad a_\theta = r\ddot{\theta} + 2\dot{r}\dot{\theta}$$

Create a subdirectory called POLC (POLar Coordinates), which we will use to calculate velocities and accelerations in polar coordinates. Within that subdirectory, enter the following variables:

| Program or value | Store into variable: |
|---|---|
| << PEQ  STEQ MINIT NAME LIST MITM MSOLVR >> | **SOLVEP** |
| "vel. & acc. polar coord." | **NAME** |
| { r rD rDD θD θDD vr vθ v ar aθ a } | **LIST** |
| { 'vr = rD'  'vθ = r*θD'   'v = √(vr^2 + vθ^2)'<br>'ar = rDD – r*θD^2'  'aθ = r*θDD + 2*rD*θD'<br>'a = √(ar^2 + aθ^2)' } | **PEQ** |

*An explanation of the variables follows*:

**SOLVEP** = a program that triggers the multiple equation solver for the particular set of equations stored in variable **PEQ**;

**NAME** = a variable storing the name of the multiple equation solver, namely, *"vel. & acc. polar coord."*;

**LIST** = a list of the variable used in the calculations, placed in the order we want them to show up in the multiple equation solver environment;

**PEQ** = list of equations to be solved, corresponding to the radial and transverse components of velocity (**vr**, **v**θ) and acceleration (**ar**, **a** ) in polar coordinates, as well as equations to calculate the magnitude of the velocity (**v**) and the acceleration (**a**) when the polar components are known.

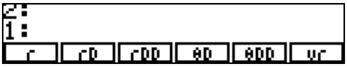
**r**, **rD**, **rDD** = r (radial coordinate), r-dot (first derivative of r), r-double dot (second derivative of r).

θ**D**, θ**DD** = θ-dot (first derivative of θ), θ-double dot (second derivative of θ).

---

Suppose you are given the following information: r = 2.5, rD = 0.5, rDD = -1.5, θD = 2.3, θDD = -6.5, and you are asked to find vr, vθ, ar, aθ, v, and a.

Start the multiple equation solver by pressing VAR SOLVE. The calculator produces a screen labeled , *"vel. & acc. polar coord."*, that looks as follows:

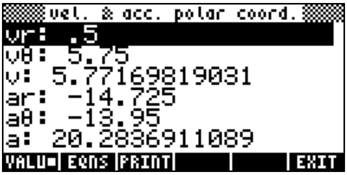

```
2:
1:
 r    rD   rDD   θD   θDD   vr
```

To enter the values of the known variables, just type the value and press the button corresponding to the variable to be entered. Use the following keystrokes: 2.5 [ r ] 0.5 [ rD ] 1.5 +/- [ rDD ] 2.3 [ θD ] 6.5 +/- [ θDD ].

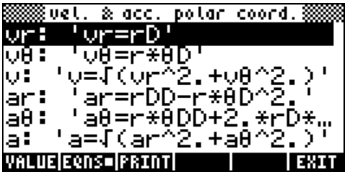
Notice that after you enter a particular value, the calculator displays the variable and its value in the upper left corner of the display. We have now entered the known variables. To calculate the unknowns we can proceed in two ways:

a). Solve for individual variables, for example, ↰[ vr ] gives vr: 0.500. Press NXT ↰[ vθ ] to get vθ : 5.750 , and so on. The remaining results are v: 5.77169819031; *a*r: -14.725; *a*θ: -13.95; and *a*: 20.2836911089.; or,

b). Solve for all variables at once, by pressing ↰ALL. The calculator will flash the solutions as it finds them. When the calculator stops, you can press ↱ALL to list all results. For this case we have:

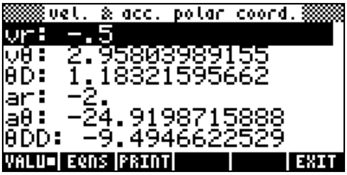

```
vel. & acc. polar coord.
vr: .5
vθ: 5.75
v: 5.77169819031
ar: -14.725
aθ: -13.95
a: 20.2836911089
VALU■ EQNS PRINT       EXIT
```

Pressing the soft-menu key EQNS will let you know the equations used to solve for each of the values in the screen:

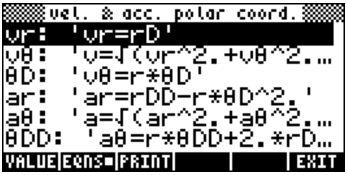

```
vel. & acc. polar coord.
vr: 'vr=rD'
vθ: 'vθ=r*θD'
v: 'v=√(vr^2.+vθ^2.)'
ar: 'ar=rDD-r*θD^2.'
aθ: 'aθ=r*θDD+2.*rD*…
a: 'a=√(ar^2.+aθ^2.)'
VALUE EQNS■ PRINT       EXIT
```

To use a new set of values press, either EXIT ALL NXT NXT, or VAR SOLVE.

Let's try another example using r = 2.5, vr = rD = -0.5, rDD = 1.5, v = 3.0, a = 25.0. Find, θD, θDD, vθ, ar, and aθ. You should get the following results:



```
vel. & acc. polar coord.
vr:  -.5
vθ:  2.95803989155
θD:  1.18321595662
ar:  -2.
aθ:  -24.9198715888
θDD:  -9.4946622529
VALU■ EQNS PRINT       EXIT
```



```
vel. & acc. polar coord.
vr:  'vr=rD'
vθ:  'v=√(vr^2.+vθ^2.…
θD:  'vθ=r*θD'
ar:  'ar=rDD-r*θD^2.'
aθ:  'a=√(ar^2.+aθ^2.…
θDD:  'aθ=r*θDD+2.*rD…
VALUE EQNS■ PRINT       EXIT
```

# Chapter 8
# Operations with lists

Lists are a type of calculator's object that can be useful for data processing and in programming. This Chapter presents examples of operations with lists.

## Definitions

A list, within the context of the calculator, is a series of objects enclosed between braces and separated by spaces (SPC), in the RPN mode, or commas (↱ ,), in both modes. Objects that can be included in a list are numbers, letters, character strings, variable names, and/or operators. Lists are useful for manipulating data sets and in some programming applications. Some examples of lists are:

```
{ t 1 }, {"BETA" h2 4},{1 1.5 2.0},
{a a a a},{ {1 2 3} {3 2 1} {1 2 3}}
```

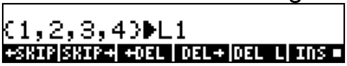
In the examples shown below we will limit ourselves to numerical lists.

## Creating and storing lists

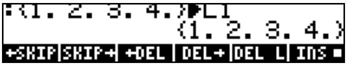
To create a list in ALG mode, first enter the braces key ↰ {} (associated with the + key), then type or enter the elements of the list, separating them with commas (↱ ,). The following keystrokes will enter the list {1 2 3 4} and store it into variable L1.

↰ {} 1 ↱ , 2 ↱ , 3 ↱ , 4
▶ STO▶ ALPHA L 1 ENTER

The screen will show the following:



```
{1,2,3,4}▶L1
```



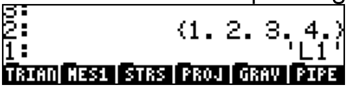
```
:{1. 2. 3. 4.}▶L1
              {1. 2. 3. 4.}
```

The figure to the left shows the screen before pressing ENTER, while the one to the right shows the screen after storing the list into L1. Notice that before pressing ENTER the list shows the commas separating its elements. However, after pressing ENTER, the commas are replaced with spaces.

Entering the same list in RPN mode requires the following keystrokes:

↰ {} 1 SPC 2 SPC 3 SPC 4 ENTER
ALPHA L 1 ENTER ▶ STO▶

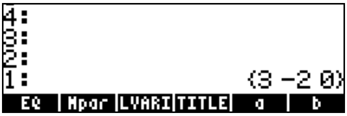
The figure below shows the RPN stack before pressing the STO▶ key:



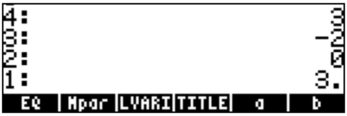
```
3:
2:           {1. 2. 3. 4.}
1:                   'L1'
```

## Composing and decomposing lists

Composing and decomposing lists makes sense in RPN mode only. Under such operating mode, decomposing a list is achieved by using function OBJ→. With this function, a list in the RPN stack is decomposed into its elements, with stack level 1: showing the number of elements in the list. The next two screen shots show the stack with a small list before and after application of function OBJ→:



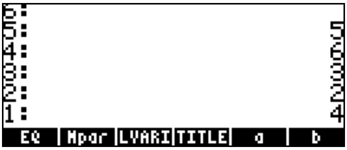
```
4:
3:
2:
1:            {3 -2 0}
```



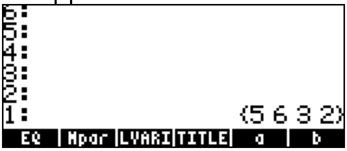
```
4:                   3
3:                  -2
2:                   0
1:                  3.
```

Notice that, after applying OBJ→, the elements of the list occupy levels 4: through 2:, while level 1: shows the number of elements in the list.

To compose a list in RPN mode, place the elements of the list in the stack, enter the list size, and apply function →LIST (select it from the function catalog, as follows: ↱ CAT ↱ ⟶ , then use the up and down arrow keys (▲ ▼) to locate function →LIST). The following screen shots show the elements of a list of size 4 before and after application of function →LIST:

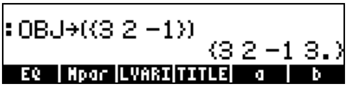


```
6:
5:                   5
4:                   6
3:                   3
2:                   2
1:                   4
```



```
6:
5:
4:
3:
2:
1:           {5 6 3 2}
```

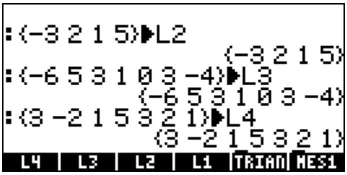
**Note:** Function OBJ→ applied to a list in ALG mode simply reproduces the list, adding to it the list size:



```
:OBJ→({3 2 -1})
             {3 2 -1 3.}
```

# Operations with lists of numbers

To demonstrate operations with lists of numbers, we will create a couple of other lists, besides list L1 created above: L2={-3,2,1,5}, L3={-6,5,3,1,0,3,-4}, L4={3,-2,1,5,3,2,1}. In ALG mode, the screen will look like this after entering lists L2, L3, L4:

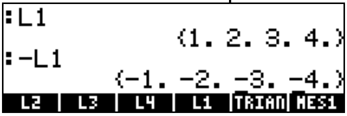


```
:{-3 2 1 5}▶L2
                 {-3 2 1 5}
:{-6 5 3 1 0 3 -4}▶L3
          {-6 5 3 1 0 3 -4}
:{3 -2 1 5 3 2 1}▶L4
           {3 -2 1 5 3 2 1}
```

In RPN mode, the following screen shows the three lists and their names ready to be stored. To store the lists in this case you need to press STO▶ three times.

## Changing sign

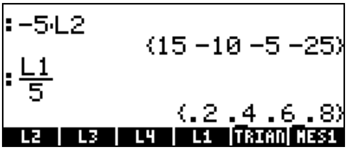
The sign-change key (+/-) , when applied to a list of numbers, will change the sign of all elements in the list. For example:



```
:L1
              {1. 2. 3. 4.}
:-L1
          {-1. -2. -3. -4.}
```

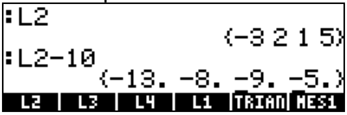
## Addition, subtraction, multiplication, division

Multiplication and division of a list by a single number is distributed across the list, for example:



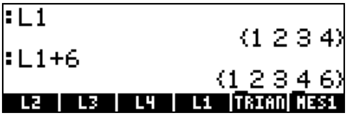
```
:-5·L2
            {15 -10 -5 -25}
: L1
  5
              {.2 .4 .6 .8}
```

Subtraction of a single number from a list will subtract the same number from each element in the list, for example:



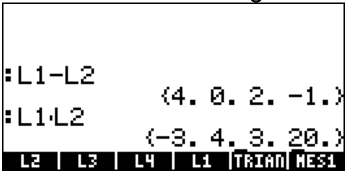
```
:L2
                 {-3 2 1 5}
:L2-10
        {-13. -8. -9. -5.}
```

Addition of a single number to a list produces a list augmented by the number, and not an addition of the single number to each element in the list. For example:

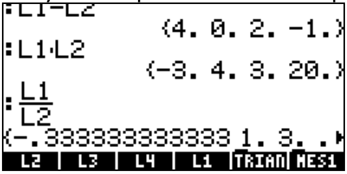


```
:L1
                    {1 2 3 4}
:L1+6
                  {1 2 3 4 6}
 L2 | L3 | L4 | L1 |TRIAN|MES1
```

Subtraction, multiplication, and division of lists of numbers of the same length produce a list of the same length with term-by-term operations. Examples:

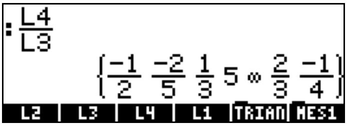


```
:L1-L2
             {4. 0. 2. -1.}
:L1·L2
            {-3. 4. 3. 20.}
 L2 | L3 | L4 | L1 |TRIAN|MES1
```



```
:L1-L2
             {4. 0. 2. -1.}
:L1·L2
            {-3. 4. 3. 20.}
:L1/L2
{-.333333333333 1. 3. .▸
 L2 | L3 | L4 | L1 |TRIAN|MES1
```

The division L4/L3 will produce an infinity entry because one of the elements in L3 is zero:



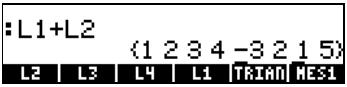
:$\frac{L4}{L3}$

$\left\{\frac{-1}{2}\ \frac{-2}{5}\ \frac{1}{3}\ 5\ \infty\ \frac{2}{3}\ \frac{-1}{4}\right\}$

L2 | L3 | L4 | L1 | TRIAN | MES1

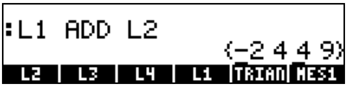
If the lists involved in the operation have different lengths, an error message is produced (Error: Invalid Dimension).

The plus sign (+), when applied to lists, acts a *concatenation* operator, putting together the two lists, rather than adding them term-by-term. For example:



```
:L1+L2
         {1 2 3 4 -3 2 1 5}
 L2 | L3 | L4 | L1 |TRIAN|MES1
```

In order to produce term-by-term addition of two lists of the same length, we need to use operator ADD. This operator can be loaded by using the function catalog (↱ CAT ). The screen below shows an application of ADD to add lists L1 and L2, term-by-term:

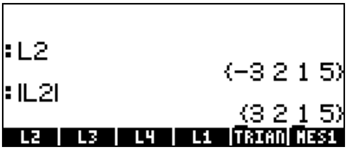


```
:L1 ADD L2
                  {-2 4 4 9}
 L2 | L3 | L4 | L1 |TRIAN|MES1
```

# Real number functions from the keyboard

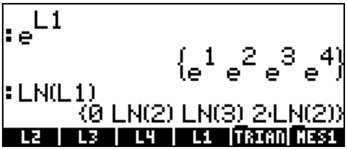
Real number functions from the keyboard (ABS, $e^x$, LN, $10^x$, LOG, SIN, $x^2$, √, COS, TAN, ASIN, ACOS, ATAN, $y^x$) can be used on lists. Here are some examples:

ABS



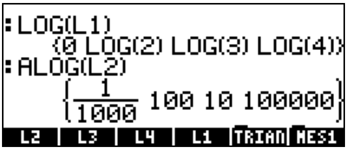
:L2
{-3 2 1 5}
:|L2|
{3 2 1 5}

EXP and LN



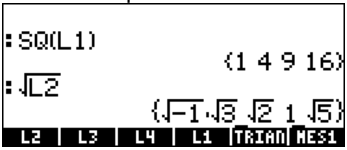
:$e^{L1}$
$\{e^1\ e^2\ e^3\ e^4\}$
:LN(L1)
{0 LN(2) LN(3) 2·LN(2)}

LOG and ANTILOG



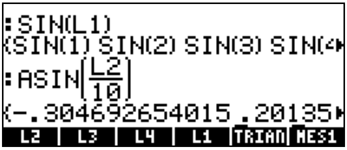
:LOG(L1)
{0 LOG(2) LOG(3) LOG(4)}
:ALOG(L2)
$\left\{\frac{1}{1000}\ 100\ 10\ 100000\right\}$

SQ and square root



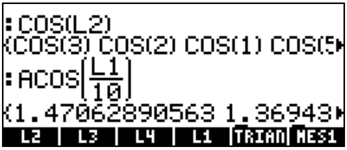
:SQ(L1)
{1 4 9 16}
:$\sqrt{L2}$
$\{\sqrt{-1}\cdot\sqrt{3}\ \sqrt{2}\ 1\ \sqrt{5}\}$

SIN, ASIN



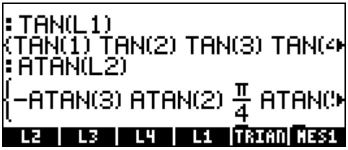
:SIN(L1)
{SIN(1) SIN(2) SIN(3) SIN(4▸
:ASIN$\left(\frac{L2}{10}\right)$
{-.304692654015 .20135▸

COS, ACOS



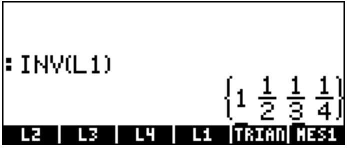
:COS(L2)
{COS(3) COS(2) COS(1) COS(5▸
:ACOS$\left(\frac{L1}{10}\right)$
{1.47062890563 1.36943▸

TAN, ATAN



:TAN(L1)
{TAN(1) TAN(2) TAN(3) TAN(4▸
:ATAN(L2)
$\{$-ATAN(3) ATAN(2) $\frac{\pi}{4}$ ATAN(▸

INVERSE (1/x)

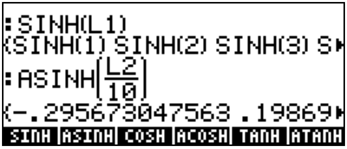


:INV(L1)
$\left\{1\ \frac{1}{2}\ \frac{1}{3}\ \frac{1}{4}\right\}$

# Real number functions from the MTH menu

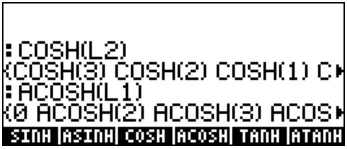
Functions of interest from the MTH menu include, from the HYPERBOLIC menu: SINH, ASINH, COSH, ACOSH, TANH, ATANH, and from the REAL menu: %, %CH, %T, MIN, MAX, MOD, SIGN, MANT, XPON, IP, FP, RND, TRNC, FLOOR, CEIL, D→R, R→D. Some of the functions that take a single argument are illustrated below applied to lists of real numbers:

SINH, ASINH



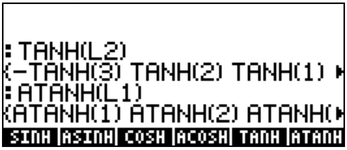
```
:SINH(L1)
{SINH(1) SINH(2) SINH(3) S▸
:ASINH(L2/10)
{-.295673047563 .19869▸
```

COSH, ACOSH



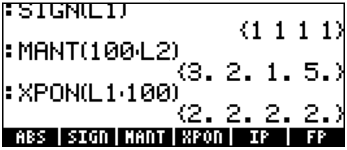
```
:COSH(L2)
{COSH(3) COSH(2) COSH(1) C▸
:ACOSH(L1)
{0 ACOSH(2) ACOSH(3) ACOS▸
```

TANH, ATANH



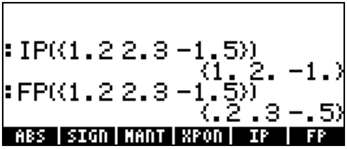
```
:TANH(L2)
{-TANH(3) TANH(2) TANH(1) ▸
:ATANH(L1)
{ATANH(1) ATANH(2) ATANH(▸
```

SIGN, MANT, XPON



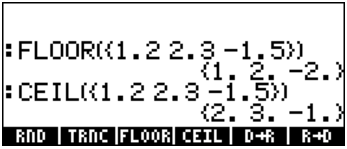
```
:SIGN(L1)
                    {1 1 1 1}
:MANT(100·L2)
               {3. 2. 1. 5.}
:XPON(L1·100)
               {2. 2. 2. 2.}
```

IP, FP



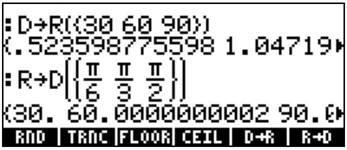
```
:IP({1.2 2.3 -1.5})
                 {1. 2. -1.}
:FP({1.2 2.3 -1.5})
                 {.2 .3 -.5}
```

FLOOR, CEIL



```
:FLOOR({1.2 2.3 -1.5})
                 {1. 2. -2.}
:CEIL({1.2 2.3 -1.5})
                 {2. 3. -1.}
```

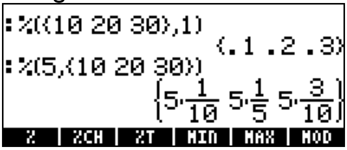
D→R, R→D



```
:D→R({30 60 90})
{.523598775598 1.04719▸
:R→D({π/6 π/3 π/2})
{30. 60.0000000002 90.0▸
```

## Examples of functions that use two arguments

The screen shots below show applications of the function % to list arguments. Function % requires two arguments. The first two examples show cases in which only one of the two arguments is a list.



```
:%({10 20 30},1)
                 {.1 .2 .3}
:%(5,{10 20 30})
       {5·1/10 5·1/5 5·3/10}
```

The results are lists with the function % distributed according to the list argument. For example,

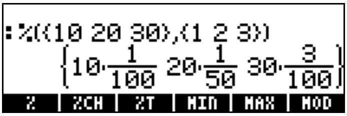
%({10, 20, 30},1) = {%(10,1),%(20,1),%(30,1)},

while

%(5,{10,20,30}) = {%(5,10),%(5,20),%(5,30)}

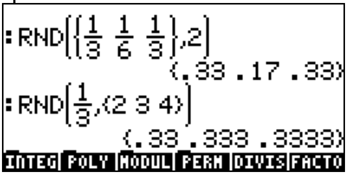
In the following example, both arguments of function % are lists of the same size. In this case, a term-by-term distribution of the arguments is performed, i.e.,

%({10,20,30},{1,2,3}) = {%(10,1),%(20,2),%(30,3)}



```
:%({10 20 30},{1 2 3})
    {10·1/100 20·1/50 30·3/100}
```

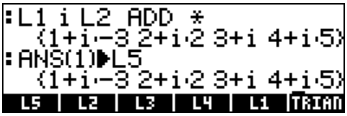
This description of function % for list arguments shows the general pattern of evaluation of any function with two arguments when one or both arguments are lists. Examples of applications of function RND are shown next:



```
:RND({1/3 1/6 1/3},2)
                {.33 .17 .33}
:RND(1/3,{2 3 4})
            {.33 .333 .3333}
```

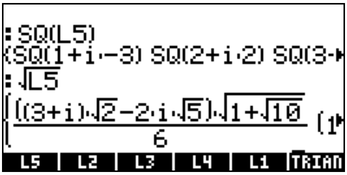
## Lists of complex numbers

The following exercise shows how to create a list of complex numbers given two lists of the same length, one representing the real parts and one the imaginary parts of the complex numbers. Use L1 ADD i*L2. The screen also shows that the resulting complex-number list is stored into variable L5:

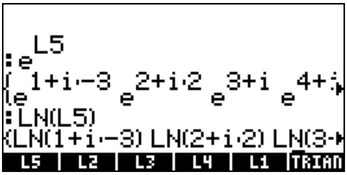


```
:L1 i L2 ADD *
  {1+i·-3 2+i·2 3+i 4+i·5}
:ANS(1)▶L5
  {1+i·-3 2+i·2 3+i 4+i·5}
```

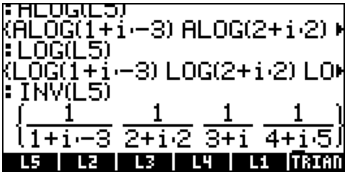
Functions such as LN, EXP, SQ, etc., can also be applied to a list of complex numbers, e.g.,



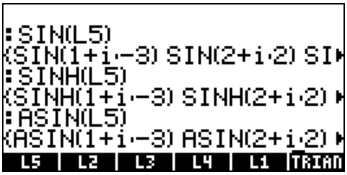
```
:SQ(L5)
{SQ(1+i·-3) SQ(2+i·2) SQ(3-▸
:√L5
{((3+i)·√2-2·i·√5)·√(1+√10)/6 (1▸
```



```
:e^L5
{e^(1+i·-3) e^(2+i·2) e^(3+i) e^(4+i▸
:LN(L5)
{LN(1+i·-3) LN(2+i·2) LN(3-▸
```

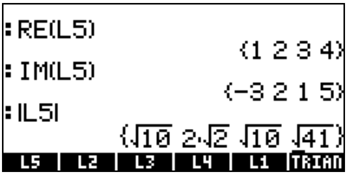


```
:ALOG(L5)
{ALOG(1+i·-3) ALOG(2+i·2) ▸
:LOG(L5)
{LOG(1+i·-3) LOG(2+i·2) LO▸
:INV(L5)
{1/(1+i·-3) 1/(2+i·2) 1/(3+i) 1/(4+i·5)}
```

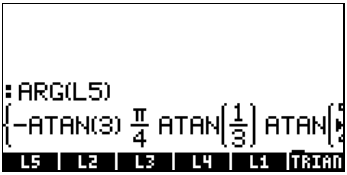


```
:SIN(L5)
{SIN(1+i·-3) SIN(2+i·2) SI▸
:SINH(L5)
{SINH(1+i·-3) SINH(2+i·2) ▸
:ASIN(L5)
{ASIN(1+i·-3) ASIN(2+i·2) ▸
```

The following example shows applications of the functions RE(Real part), IM(imaginary part), ABS(magnitude), and ARG(argument) of complex numbers. The results are lists of real numbers:



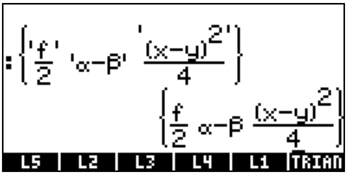
```
:RE(L5)
                {1 2 3 4}
:IM(L5)
               {-3 2 1 5}
:|L5|
       {√10 2·√2 √10 √41}
```



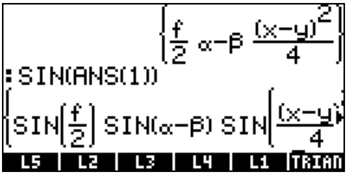
```
:ARG(L5)
{-ATAN(3) π/4 ATAN(1/3) ATAN(▸
```

## Lists of algebraic objects

The following are examples of lists of algebraic objects with the function SIN applied to them:



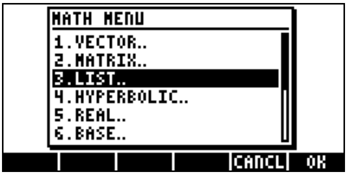
```
:{'f/2' 'α-β' '(x-y)^2/4'}
              {f/2 α-β (x-y)^2/4}
```

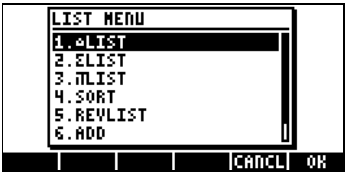


```
              {f/2 α-β (x-y)^2/4}
:SIN(ANS(1))
{SIN(f/2) SIN(α-β) SIN((x-y▸/4
```

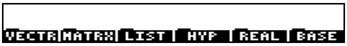
## The MTH/LIST menu

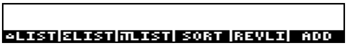
The MTH menu provides a number of functions that exclusively to lists. With flag 117 set to CHOOSE boxes:





Next, with system flag 117 set to SOFT menus:

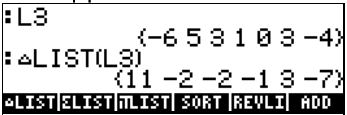




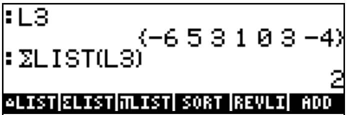
This menu contains the following functions:

ΔLIST : Calculate increment among consecutive elements in list
ΣLIST : Calculate summation of elements in the list
ΠLIST : Calculate product of elements in the list
SORT : Sorts elements in increasing order
REVLIST : Reverses order of list
ADD : Operator for term-by-term addition of two lists of the same length (examples of this operator were shown above)

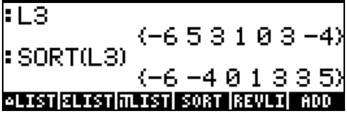
Examples of application of these functions in ALG mode are shown next:



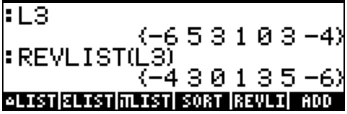
```
:L3
          {-6 5 3 1 0 3 -4}
:ΔLIST(L3)
         {11 -2 -2 -1 3 -7}
```



```
:L3
          {-6 5 3 1 0 3 -4}
:ΣLIST(L3)
                          2
```

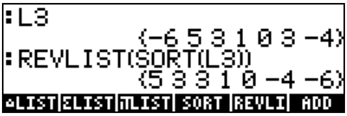


```
:L3
          {-6 5 3 1 0 3 -4}
:SORT(L3)
          {-6 -4 0 1 3 3 5}
```



```
:L3
          {-6 5 3 1 0 3 -4}
:REVLIST(L3)
          {-4 3 0 1 3 5 -6}
```

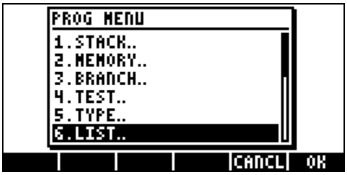
SORT and REVLIST can be combined to sort a list in decreasing order:

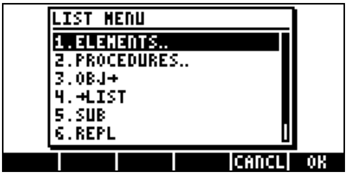


```
:L3
          {-6 5 3 1 0 3 -4}
:REVLIST(SORT(L3))
          {5 3 3 1 0 -4 -6}
```

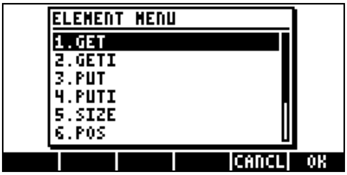
## Manipulating elements of a list

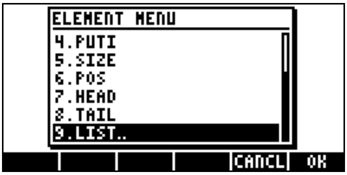
The PRG (programming) menu includes a LIST sub-menu with a number of functions to manipulate elements of a list. With system flag 117 set to CHOOSE boxes:





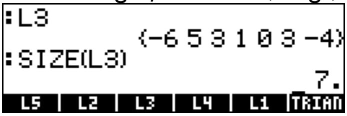
Item 1. ELEMENTS.. contains the following functions that can be used for the manipulation of elements in lists:





### List size

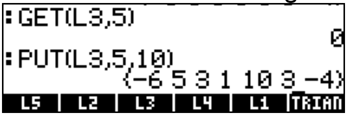
Function SIZE, from the PRG/LIST/ELEMENTS sub-menu, can be used to obtain the size (also known as length) of the list, e.g.,



```
:L3
                {-6 5 3 1 0 3 -4}
:SIZE(L3)
                               7.
```

### Extracting and inserting elements in a list

To extract elements of a list we use function GET, available in the PRG/LIST/ELEMENTS sub-menu. The arguments of function GET are the list and the number of the element you want to extract. To insert an element into a list use function PUT (also available in the PRG/LST/ELEMENTS sub-menu). The arguments of function PUT are the list, the position that one wants to replace, and the value that will be replaced. Examples of applications of functions GET and PUT are shown in the following screen:

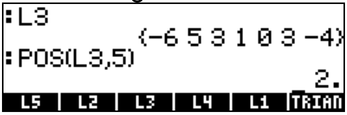


```
:GET(L3,5)
                                0
:PUT(L3,5,10)
             {-6 5 3 1 10 3 -4}
```

Functions GETI and PUTI, also available in sub-menu PRG/ ELEMENTS/, can also be used to extract and place elements in a list. These two functions, however, are useful mainly in programming. Function GETI uses the same arguments as GET and returns the list, the element location plus one, and the element at the location requested. Function PUTI uses the same arguments as GET and returns the list and the list size.

## Element position in the list

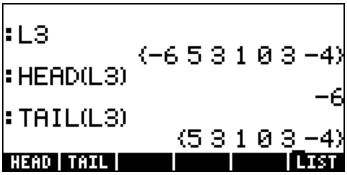
To determine the position of an element in a list use function POS having the list and the element of interest as arguments. For example,



```
:L3
              {-6 5 3 1 0 3 -4}
:POS(L3,5)
                              2.
```

## HEAD and TAIL functions

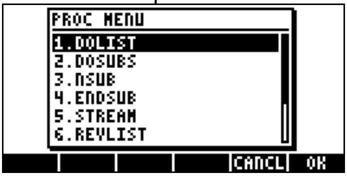
The HEAD function extracts the first element in the list. The TAIL function removes the first element of a list, returning the remaining list. Some examples are shown next:

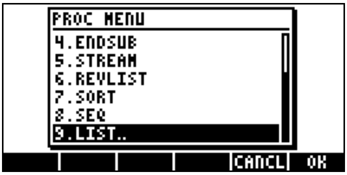


```
:L3
              {-6 5 3 1 0 3 -4}
:HEAD(L3)
                              -6
:TAIL(L3)
                 {5 3 1 0 3 -4}
```

## The SEQ function

Item 2. PROCEDURES.. in the PRG/LIST menu contains the following functions that can be used to operate on lists.



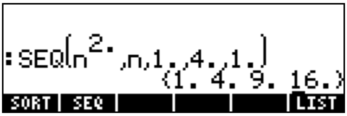


Functions REVLIST and SORT were introduced earlier as part of the MTH/LIST menu. Functions DOLIST, DOSUBS, NSUB, ENDSUB, and STREAM, are designed as programming functions for operating lists in RPN mode. Function

SEQ is useful to produce a list of values given a particular expression and is described in more detail here.

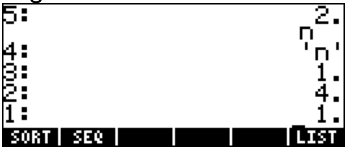
The SEQ function takes as arguments an expression in terms of an index, the name of the index, and starting, ending, and increment values for the index, and returns a list consisting of the evaluation of the expression for all possible values of the index. The general form of the function is SEQ(*expression, index, start, end, increment*).

In the following example, in ALG mode, we identify *expression* = $n^2$, *index* = n, *start* = 1, *end* = 4, and *increment* = 1:



```
:SEQ(n^2.,n,1.,4.,1.)
                {1. 4. 9. 16.}
```

The list produced corresponds to the values {$1^2$, $2^2$, $3^2$, $4^2$}. In RPN mode, you can list the different arguments of the function as follows:

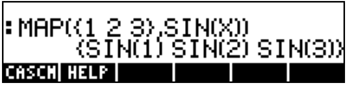


```
5:                          n^2.
4:                           'n'
3:                            1.
2:                            4.
1:                            1.
```

before applying function SEQ.

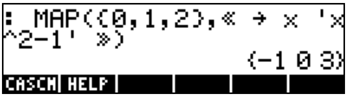
## The MAP function

The MAP function, available through the command catalog (→ CAT ), takes as arguments a list of numbers and a function f(X) or a program of the form << → a … >>, and produces a list consisting of the application of function f or the program to the list of numbers. For example, the following call to function MAP applies the function SIN(X) to the list {1,2,3}:



```
:MAP({1 2 3},SIN(X))
     {SIN(1) SIN(2) SIN(3)}
```

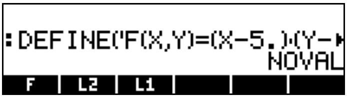
The following call to function MAP uses a program instead of a function as second argument:



```
: MAP({0,1,2},« → x 'x
^2-1' »)
                    {-1 0 3}
```

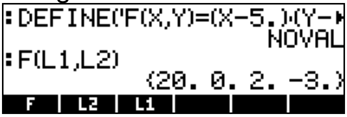
## Defining functions that use lists

In Chapter 3 we introduced the use of the DEFINE function ( ↰ DEF ) to create functions of real numbers with one or more arguments. A function defined with DEF can also be used with list arguments, except that, any function incorporating an addition must use the ADD operator rather than the plus sign (+). For example, if we define the function F(X,Y) = (X-5)*(Y-2), shown here in ALG mode:



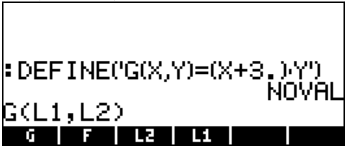
```
:DEFINE('F(X,Y)=(X-5.)·(Y-▸
                       NOVAL
```

we can use lists (e.g., variables L1 and L2, defined earlier in this Chapter) to evaluate the function, resulting in:

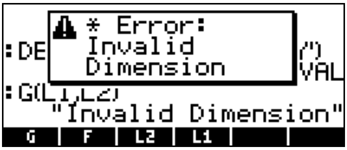


```
:DEFINE('F(X,Y)=(X-5.)·(Y-▸
                       NOVAL
:F(L1,L2)
            {20. 0. 2. -3.}
```

Since the function statement includes no additions, the application of the function to list arguments is straightforward. However, if we define the function G(X,Y) = (X+3)*Y, an attempt to evaluate this function with list arguments (L1, L2) will fail:

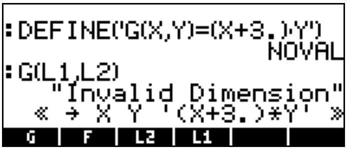


```
:DEFINE('G(X,Y)=(X+3.)·Y')
                       NOVAL
G(L1,L2)
```



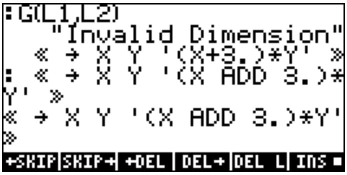
```
     * Error:
       Invalid
       Dimension
:G(L1,L2)
   "Invalid Dimension"
```

To fix this problem we can edit the contents of variable G , which we can list in the stack by using ↱ G,



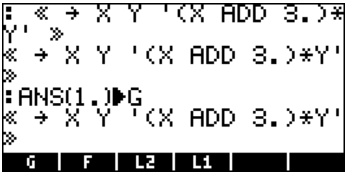
```
:DEFINE('G(X,Y)=(X+3.)·Y')
                       NOVAL
:G(L1,L2)
   "Invalid Dimension"
 « → X Y '(X+3.)*Y' »
```

to replace the plus sign (+) with ADD:



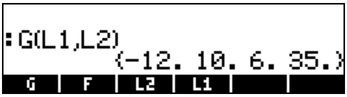
```
:G(L1,L2)
   "Invalid Dimension"
  « → X Y '(X+3.)*Y' »
: « → X Y '(X ADD 3.)*
Y' »
« → X Y '(X ADD 3.)*Y'
»
```

Next, we store the edited expression into variable G:



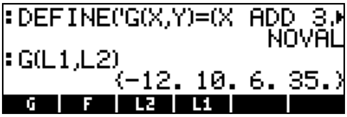
```
: « → X Y '(X ADD 3.)*
Y' »
« → X Y '(X ADD 3.)*Y'
»
:ANS(1.)▶G
« → X Y '(X ADD 3.)*Y'
»
```

Evaluating G(L1,L2) now produces the following result:



```
:G(L1,L2)
        {-12. 10. 6. 35.}
```

As an alternative, you can define the function with ADD rather than the plus sign (+), from the start, i.e., use `DEFINE('G(X,Y)=(X ADD 3)*Y')` :



```
:DEFINE('G(X,Y)=(X ADD 3.▶
                    NOVAL
:G(L1,L2)
        {-12. 10. 6. 35.}
```

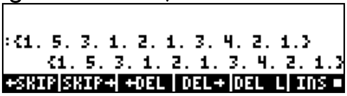
You can also define the function as G(X,Y) = (X--3)*Y.

## Applications of lists

This section shows a couple of applications of lists to the calculation of statistics of a sample. By a sample we understand a list of values, say, {$s_1$, $s_2$, …, $s_n$}. Suppose that the sample of interest is the list

```
{1, 5, 3, 1, 2, 1, 3, 4, 2, 1}
```

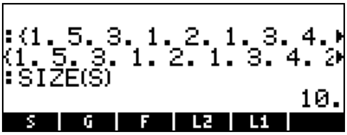
and that we store it into a variable called S (The screen shot below shows this action in ALG mode, however, the procedure in RPN mode is very similar. Just keep in mind that in RPN mode you place the arguments of functions in the stack before activating the function):



```
:{1. 5. 3. 1. 2. 1. 3. 4. 2. 1.}
    {1. 5. 3. 1. 2. 1. 3. 4. 2. 1.}
```

## Harmonic mean of a list

This is a small enough sample that we can count on the screen the number of elements (n=10). For a larger list, we can use function SIZE to obtain that number, e.g.,



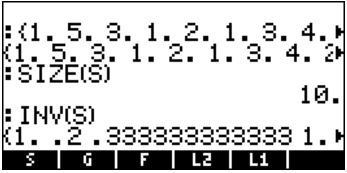
```
:{1. 5. 3. 1. 2. 1. 3. 4.▸
{1. 5. 3. 1. 2. 1. 3. 4. 2▸
:SIZE(S)
                       10.
```

Suppose that we want to calculate the harmonic mean of the sample, defined as

$$s_h = \frac{1}{\frac{1}{n}\sum_{k=1}^{n}\frac{1}{s_n}} = \frac{1}{\frac{1}{n}\left(\frac{1}{s_1}+\frac{1}{s_2}+\cdots+\frac{1}{s_n}\right)}.$$

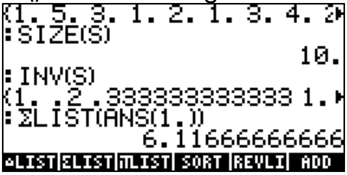
To calculate this value we can follow this procedure:

1. Apply function INV () to list S:



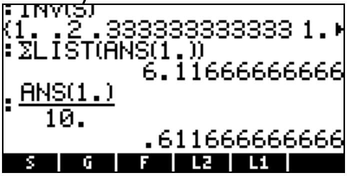
```
:{1. 5. 3. 1. 2. 1. 3. 4.▸
{1. 5. 3. 1. 2. 1. 3. 4. 2▸
:SIZE(S)
                       10.
:INV(S)
{1. .2 .333333333333 1.▸
```

2. Apply function ΣLIST() to the resulting list in1.



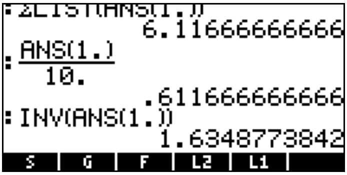
```
{1. 5. 3. 1. 2. 1. 3. 4. 2▸
:SIZE(S)
                       10.
:INV(S)
{1. .2 .333333333333 1.▸
:ΣLIST(ANS(1.))
             6.11666666666
```

3. Divide the result above by n = 10:



```
:INV(S)
{1. .2 .333333333333 1.▸
:ΣLIST(ANS(1.))
              6.11666666666
:ANS(1.)
   10.
             .611666666666
 S | G | F | L2 | L1 |
```

4. Apply the INV() function to the latest result:



```
:ΣLIST(ANS(1.))
              6.11666666666
:ANS(1.)
   10.
             .611666666666
:INV(ANS(1.))
               1.6348773842
 S | G | F | L2 | L1 |
```

Thus, the harmonic mean of list S is $s_h$ = 1.6348…

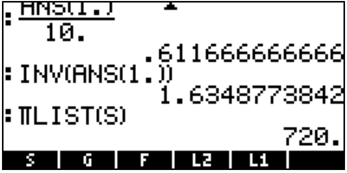
## Geometric mean of a list

The geometric mean of a sample is defined as

$$x_g = \sqrt[n]{\prod_{k=1}^{n} x_k} = \sqrt[n]{x_1 \cdot x_2 \cdots x_n}$$

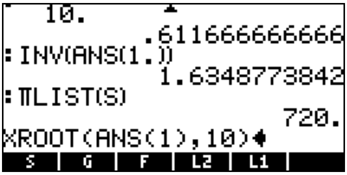
To find the geometric mean of the list stored in S, we can use the following procedure:

1. Apply function ΠLIST() to list S:

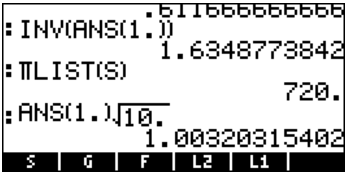


```
:ANS(1.)          ▲
   10.
             .611666666666
:INV(ANS(1.))
               1.6348773842
:ΠLIST(S)
                       720.
 S | G | F | L2 | L1 |
```

2. Apply function XROOT(x,y), i.e., keystrokes ↱ ˣ√y , to the result in 1:



```
   10.            ▲
             .611666666666
:INV(ANS(1.))
               1.6348773842
:ΠLIST(S)
                       720.
XROOT(ANS(1),10)◆
 S | G | F | L2 | L1 |
```



```
             .611666666666
:INV(ANS(1.))
               1.6348773842
:ΠLIST(S)
                       720.
:ANS(1.)√10.
               1.00320315402
 S | G | F | L2 | L1 |
```

Thus, the geometric mean of list S is $s_g$ = 1.003203…

## Weighted average

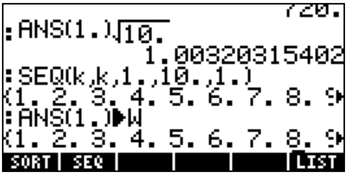
Suppose that the data in list S, defined above, namely:

S = {1,5,3,1,2,1,3,4,2,1}

is affected by the weights,

W = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

If we define the weight list as W = {$w_1$,$w_2$,…,$w_n$}, we notice that the *k* -th element in list W, above, can be defined by $w_k$ = k. Thus we can use function SEQ to generate this list, and then store it into variable ▓ as follows:



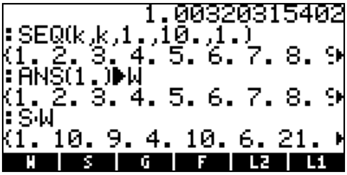
```
                    720.
:ANS(1.)√10.
          1.00320315402
:SEQ(k,k,1.,10.,1.)
{1. 2. 3. 4. 5. 6. 7. 8. 9▸
:ANS(1.)▶W
{1. 2. 3. 4. 5. 6. 7. 8. 9▸
```

Given the data list {$s_1$, $s_2$, …, $s_n$ }, and the weight list {$w_1$, $w_2$, …, $w_n$ }, the weighted average of the data in S is defined as

$$s_w = \frac{\sum_{k=1}^{n} w_k \cdot s_k}{\sum_{k=1}^{n} w_k}.$$

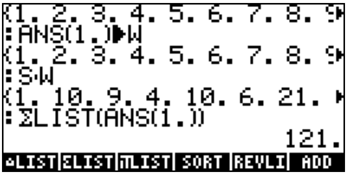
To calculate the weighted average of the data in list S with the weights in list W, we can use th e following steps:

1. Multiply lists S and W:



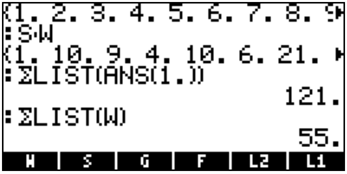
```
          1.00320315402
:SEQ(k,k,1.,10.,1.)
{1. 2. 3. 4. 5. 6. 7. 8. 9▸
:ANS(1.)▶W
{1. 2. 3. 4. 5. 6. 7. 8. 9▸
:S·W
{1. 10. 9. 4. 10. 6. 21. ▸
```

2. Use function ΣLIST in this result to calculate the numerator of $s_w$:



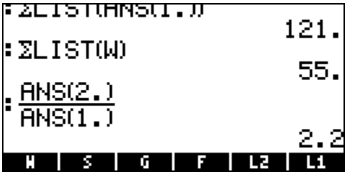
```
{1. 2. 3. 4. 5. 6. 7. 8. 9▸
:ANS(1.)▶W
{1. 2. 3. 4. 5. 6. 7. 8. 9▸
:S·W
{1. 10. 9. 4. 10. 6. 21. ▸
:ΣLIST(ANS(1.))
                    121.
```

3. Use function ΣLIST, once more, to calculate the denominator of $s_w$:



```
{1. 2. 3. 4. 5. 6. 7. 8. 9▸
:S·W
{1. 10. 9. 4. 10. 6. 21. ▸
:ΣLIST(ANS(1.))
                    121.
:ΣLIST(W)
                     55.
 W | S | G | F | L2 | L1
```

4. Use the expression ANS(2)/ANS(1) to calculate the weighted average:



```
:ΣLIST(ANS(1.))
                    121.
:ΣLIST(W)
                     55.
 ANS(2.)
:───────
 ANS(1.)
                     2.2
 W | S | G | F | L2 | L1
```

Thus, the weighted average of list S with weights in list W is $s_w$= 2.2.

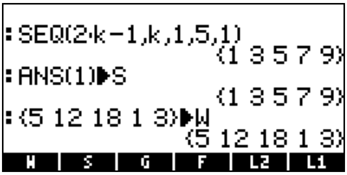
**Note:** ANS(1) refers to the most recent result (55), while ANS(2) refers to the previous to last result (121).

## Statistics of grouped data

Grouped data is typically given by a table showing the frequency (w) of data in data classes or bins. Each class or bin is represented by a class mark (s), typically the midpoint of the class. An example of grouped data is shown next:

|  | Class | Frequency |
|---|---|---|
| Class | mark | count |
| boundaries | $s_k$ | $w_k$ |
| 0 - 2 | 1 | 5 |
| 2 - 4 | 3 | 12 |
| 4 - 6 | 5 | 18 |
| 6 - 8 | 7 | 1 |
| 8 -10 | 9 | 3 |

The class mark data can be stored in variable S, while the frequency count can be stored in variable W, as follows:



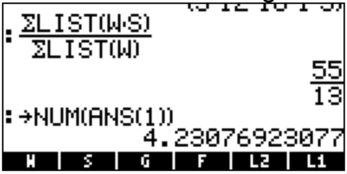
```
:SEQ(2·k-1,k,1,5,1)
                        {1 3 5 7 9}
:ANS(1)▶S
                        {1 3 5 7 9}
:{5 12 18 1 3}▶W
                     {5 12 18 1 3}
```

Given the list of class marks S = {$s_1$, $s_2$, …, $s_n$ }, and the list of frequency counts W = {$w_1$, $w_2$, …, $w_n$ }, the weighted average of the data in S with weights W represents the mean value of the grouped data, that we call $\overline{s}$, in this context:

$$\overline{s} = \frac{\sum_{k=1}^{n} w_k \cdot s_k}{\sum_{k=1}^{n} w_k} = \frac{\sum_{k=1}^{n} w_k \cdot s_k}{N},$$

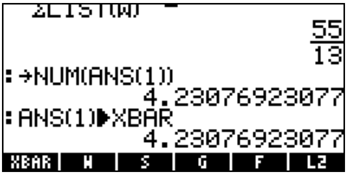
where $N = \sum_{k=1}^{n} w_k$ represents the total frequency count.

The mean value for the data in lists S and W, therefore, can be calculated using the procedure outlined above for the weighted average, i.e.,



```
: ΣLIST(W·S)
  ──────────
   ΣLIST(W)
                                 55
                                 ──
                                 13
:→NUM(ANS(1))
                      4.2307692307
```

We'll store this value into a variable called XBAR:

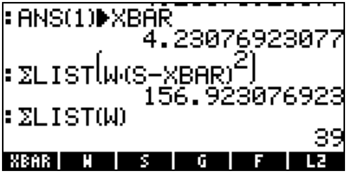


```
   ΣLIST(W)
                                 55
                                 ──
                                 13
:→NUM(ANS(1))
                      4.2307692307
:ANS(1)▶XBAR
                      4.2307692307
```

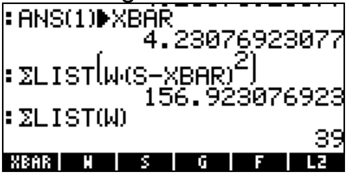
The variance of this grouped data is defined as

$$V = \frac{\sum_{k=1}^{n} w_k \cdot (s_k - \overline{s})^2}{\sum_{k=1}^{n} w_k} = \frac{\sum_{k=1}^{n} w_k \cdot (s_k - \overline{s})^2}{N}$$

To calculate this last result, we can use the following:

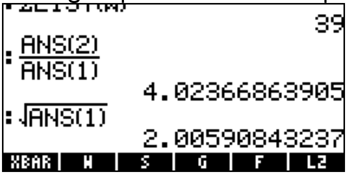


```
:ANS(1)▶XBAR
          4.23076923077
:ΣLIST(W·(S-XBAR)²)
          156.923076923
:ΣLIST(W)
                     39
```



```
:ANS(1)▶XBAR
          4.23076923077
:ΣLIST(W·(S-XBAR)²)
          156.923076923
:ΣLIST(W)
                     39
```

The standard deviation of the grouped data is the square root of the variance:



```
                     39
:ANS(2)/ANS(1)
          4.02366863905
:√ANS(1)
          2.00590843237
```

# Chapter 9
# Vectors

This Chapter provides examples of entering and operating with vectors, both mathematical vectors of many elements, as well as physical vectors of 2 and 3 components.

## Definitions

From a mathematical point of view, a vector is an array of 2 or more elements arranged into a row or a column. These will be referred to as *row* and *column vectors*. Examples are shown below:

$$v = \begin{bmatrix} -1 \\ 3 \\ 6 \end{bmatrix}, \quad u = [1, -3, 5, 2]$$

Physical vectors have two or three components and can be used to represent physical quantities such as position, velocity, acceleration, forces, moments, linear and angular momentum, angular velocity and acceleration, etc. Referring to a Cartesian coordinate system (x,y,z), there exists unit vectors **i**, **j**, **k** associated with each coordinate direction, such that a physical vector **A** can be written in terms of its components $A_x$, $A_y$, $A_z$, as **A** = $A_x$**i** + $A_y$**j** + $A_z$**k**. Alternative notation for this vector are: **A** = [$A_x$, $A_y$, $A_z$], **A** = ($A_x$, $A_y$, $A_z$), or **A** = < $A_x$, $A_y$, $A_z$ >. A two dimensional version of this vector will be written as **A** = $A_x$**i** + $A_y$**j**, **A** = [$A_x$, $A_y$], **A** = ($A_x$, $A_y$), or **A** = < $A_x$, $A_y$ >. Since in the calculator vectors are written between brackets [ ], we will choose the notation **A** = [$A_x$, $A_y$, $A_z$] or **A** = [$A_x$, $A_y$, $A_z$], to refer to two- and three-dimensional vectors from now on. The magnitude of a vector **A** is defined as |**A**| = $\sqrt{A_x^2 + A_y^2 + A_z^2}$ . A unit vector in the direction of vector **A**, is defined as $\mathbf{e}_A$ = **A**/|**A**|. Vectors can be multiplied by a scalar, e.g., k**A** = [k$A_x$, k$A_y$, k$A_z$]. Physically, the vector k**A** is parallel to vector **A**, if k>0, or anti-parallel to vector **A**, if k<0. The negative of a vector is defined as –**A** = (–1)**A** = [–$A_x$, –$A_y$, –$A_z$]. Division by as scalar can be interpreted as a multiplication, i.e., **A**/k = (1/k)·**A**. Addition and subtraction of vectors are defined as **A**±**B** = [$A_x \pm B_x$, $A_y \pm B_y$, $A_z \pm B_y$], where **B** is the vector **B** = [$B_x$, $B_y$, $B_z$].

There are two definitions of products of physical vectors, a scalar or internal product (the dot product) and a vector or external product (the cross product). The dot product produces a scalar value defined as $\mathbf{A}\bullet\mathbf{B} = |\mathbf{A}||\mathbf{B}|\cos(\theta)$, where $\theta$ is the angle between the two vectors. The cross product produces a vector $\mathbf{A}\times\mathbf{B}$ whose magnitude is $|\mathbf{A}\times\mathbf{B}| = |\mathbf{A}||\mathbf{B}|\sin(\theta)$, and its direction is given by the so-called right-hand rule (consult a textbook on Math, Physics, or Mechanics to see this operation illustrated graphically). In terms of Cartesian components, $\mathbf{A}\bullet\mathbf{B} = A_xB_x+A_yB_y+A_zB_z$, and $\mathbf{A}\times\mathbf{B} = [A_yB_z-A_zB_y,A_zB_x-A_xB_z,A_xB_y-A_yB_x]$. The angle between two vectors can be found from the definition of the dot product as $\cos(\theta) = \mathbf{A}\bullet\mathbf{B}/|\mathbf{A}||\mathbf{B}|= \mathbf{e}_A\bullet\mathbf{e}_B$. Thus, if two vectors **A** and **B** are perpendicular ($\theta = 90^0 = \pi/2^{rad}$), $\mathbf{A}\bullet\mathbf{B} = 0$.

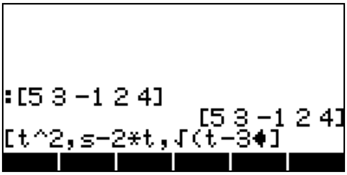
## Entering vectors

In the calculator, vectors are represented by a sequence of numbers enclosed between brackets, and typically entered as row vectors. The brackets are generated in the calculator by the keystroke combination ↰ [] , associated with the × key. The following are examples of vectors in the calculator:

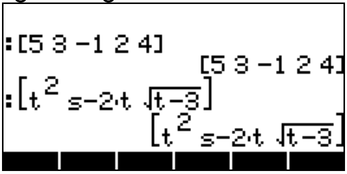
| | |
|---|---|
| `[3.5, 2.2, -1.3, 5.6, 2.3]` | A general row vector |
| `[1.5,-2.2]` | A 2-D vector |
| `[3,-1,2]` | A 3-D vector |
| `['t','t^2','SIN(t)']` | A vector of algebraics |

### Typing vectors in the stack

With the calculator in ALG mode, a vector is typed into the stack by opening a set of brackets (↰ [] ) and typing the components or elements of the vector separated by commas (↱ , ). The screen shots below show the entering of a numerical vector followed by an algebraic vector. The figure to the left shows the algebraic vector before pressing ↵. The figure to the right shows the calculator's screen after entering the algebraic vector:



```
:[5 3 -1 2 4]
                    [5 3 -1 2 4]
[t^2,s-2*t,√(t-3◆]
```



:[5 3 -1 2 4]

[5 3 -1 2 4]

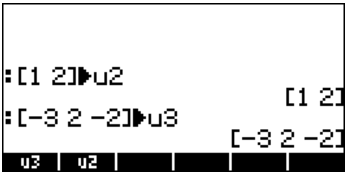
:$[t^2\ \ s-2\cdot t\ \ \sqrt{t-3}]$

$[t^2\ \ s-2\cdot t\ \ \sqrt{t-3}]$

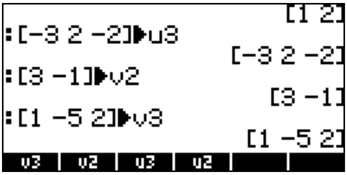
In RPN mode, you can enter a vector in the stack by opening a set of brackets and typing the vector components or elements separated by either commas (↱ ,) or spaces (SPC). Notice that after pressing ENTER , in either mode, the calculator shows the vector elements separated by spaces.

## Storing vectors into variables

Vectors can be stored into variables. The screen shots below show the vectors $\mathbf{u}_2$ = [1, 2], $\mathbf{u}_3$ = [-3, 2, -2], $\mathbf{v}_2$ = [3,-1], $\mathbf{v}_3$ = [1, -5, 2] stored into variables u2, u3, v2, and v3, respectively. First, in ALG mode:

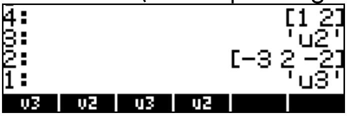


```
:[1 2]▶u2
                  [1 2]
:[-3 2 -2]▶u3
              [-3 2 -2]
```

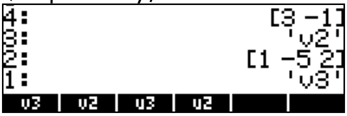


```
                  [1 2]
:[-3 2 -2]▶u3
              [-3 2 -2]
:[3 -1]▶v2
                 [3 -1]
:[1 -5 2]▶v3
               [1 -5 2]
```

Then, in RPN mode (before pressing STO▶, repeatedly):



```
4:                [1 2]
3:                 'u2'
2:            [-3 2 -2]
1:                 'u3'
```



```
4:               [3 -1]
3:                 'v2'
2:             [1 -5 2]
1:                 'v3'
```

## Using the matrix writer (MTRW) to enter vectors

Vectors can also be entered by using the matrix writer ↰ MTRW (third key in the fourth row of keys from the top of the keyboard). This command generates a species of spreadsheet corresponding to rows and columns of a matrix (Details on using the matrix writer to enter matrices will be presented in a subsequent chapter). For a vector we are interested in filling only elements in the top row. By default, the cell in the top row and first column is selected. At the bottom of the spreadsheet you will find the following soft menu keys:

EDIT VEC■ ←WID WID→ GO→■ GO↓

The EDIT key is used to edit the contents of a selected cell in the matrix writer.
The VEC key, when selected, will produce a vector, as opposite to a matrix of one row and many columns.

**Vectors vs. matrices**

To see the VEC key in action, try the following exercises:

(1) Launch the matrix writer (← MTRW ). With VEC■ and GO→■ selected, enter 3 ENTER 5 ENTER 2 ENTER ENTER . This produces [3. 5. 2.]. (In RPN mode, you can use the following keystroke sequence to produce the same result: 3 SPC 5 SPC 2 ENTER ENTER ).

(2) With VEC deselected and GO→■ selected,, enter 3 SPC 5 SPC 2 ENTER ENTER . This produces [[3. 5. 2.]].

Although these two results differ only in the number of brackets used, for the calculator they represent different mathematical objects. The first one is a vector with three elements, and the second one a matrix with one row and three columns. There are differences in the way that mathematical operations take place on a vector as opposite to a matrix. Therefore, for the time being, keep the soft menu key VEC■ selected while using the matrix writer.

The ←WID key is used to decrease the width of the columns in the spreadsheet. Press this key a couple of times to see the column width decrease in your matrix writer.

The WID→ key is used to increase the width of the columns in the spreadsheet. Press this key a couple of times to see the column width increase in your matrix writer.

The GO→■ key, when selected, automatically selects the next cell to the right of the current cell when you press ENTER . This option is selected by default.

The GO↓ key, when selected, automatically selects the next cell below the current cell when you press ENTER .

**Moving to the right vs. moving down in the matrix writer**

Activate the matrix writer and enter 3 ENTER 5 ENTER 2 ENTER ENTER with the GO→■ key selected (default). Next, enter the same sequence of numbers with the GO↓■ key selected to see the difference. In the first case you entered a vector of three elements. In the second case you entered a matrix of three rows and one column.

Activate the matrix writer again by using ← MTRW , and press NXT to check out the second soft key menu at the bottom of the display. It will show the keys:

+ROW -ROW +COL -COL →STK GOTO

The +ROW key will add a row full of zeros at the location of the selected cell of the spreadsheet.
The -ROW key will delete the row corresponding to the selected cell of the spreadsheet.
The +COL key will add a column full of zeros at the location of the selected cell of the spreadsheet.
The -COL key will delete the column corresponding to the selected cell of the spreadsheet.
The →STK key will place the contents of the selected cell on the stack.
The GOTO key, when pressed, will request that the user indicate the number of the row and column where he or she wants to position the cursor.

Pressing NXT once more produces the last menu, which contains only one function DEL (delete).

The function DEL will delete the contents of the selected cell and replace it with a zero.

To see these keys in action try the following exercise:

(1) Activate the matrix writer by using ← MTRW . Make sure the VEC■ and GO→■ keys are selected.
(2) Enter the following:

1 ENTER 2 ENTER 3 ENTER
NXT GOTO 2 OK 1 OK OK
2 ENTER 1 ENTER 5 ENTER
4 ENTER 5 ENTER 6 ENTER
7 ENTER 8 ENTER 9 ENTER

(3) Move the cursor up two positions by using ▲▲▲. Then press -ROW. The second row will disappear.
(4) Press +ROW. A row of three zeroes appears in the second row.

(5) Press -COL. The first column will disappear.
(6) Press +COL. A row of two zeroes appears in the first row.
(7) Press GOTO 3 OK 3 OK OK to move to position (3,3).
(8) Press ←STK. This will place the contents of cell (3,3) on the stack, although you will not be able to see it yet.
(9) Press ENTER to return to normal display. Element (3,3) and the full matrix will be available in the screen.

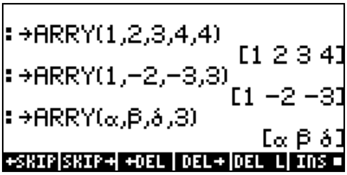
**Summary of Matrix Writer use for entering vectors**
In summary, to enter a vector using the matrix writer, simply activate the writer (← MTRW ), and place the elements of the vector, pressing ENTER after each of them. Then, press ENTER ENTER. Make sure that the VEC■ and GO→■ keys are selected.

Example: ← MTRW ' ALPHA ← X Y^x 2 ENTER 2 ENTER 5 +/- ENTER ENTER

produces: ['x^2' 2 –5 ]

## Building a vector with →ARRY

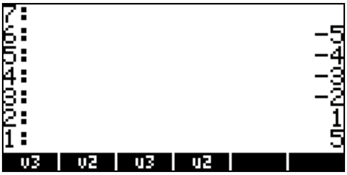
The function →ARRY, available in the function catalog (→ CAT → ⟶, use ▲▼ to locate the function), can also be used to build a vector or array in the following way. In ALG mode, enter →ARRY(*vector elements, number of elements*), e.g.,



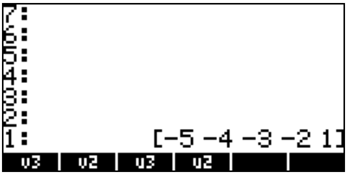
```
:→ARRY(1,2,3,4,4)
                  [1 2 3 4]
:→ARRY(1,-2,-3,3)
                 [1 -2 -3]
:→ARRY(α,β,δ,3)
                    [α β δ]
```

In RPN mode:
(1) Enter the n elements of the array in the order you want them to appear in the array (when read from left to right) into the RPN stack.
(2) Enter *n* as the last entry.
(3) Use function →ARRY.

The following screen shots show the RPN stack before and after applying function →ARRY:



```
7:
6:                -5
5:                -4
4:                -3
3:                -2
2:                 1
1:                 5
```



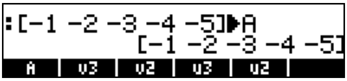
```
7:
6:
5:
4:
3:
2:
1:    [-5 -4 -3 -2 1]
```

In RPN mode, the function [→ARRY] takes the objects from stack levels *n+1, n, n-1*, …, down to stack levels 3 and 2, and converts them into a vector of *n* elements. The object originally at stack level n+1 becomes the first element, the object originally at level n becomes the second element, and so on.

**Note**: Function →ARRY is also available in the PRG/TYPE menu (← PRG )

## Identifying, extracting, and inserting vector elements

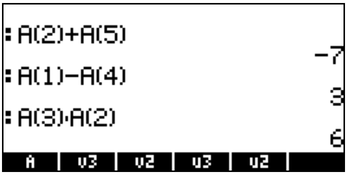
If you store a vector into a variable name, say A, you can identify elements of the vector by using A(i), where i is an integer number less than or equal to the vector size. For example, create the following array and store it in variable A: [-1, -2, -3, -4, -5]:



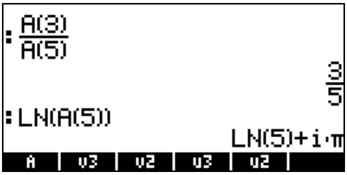
```
:[-1 -2 -3 -4 -5]▶A
          [-1 -2 -3 -4 -5]
```

To recall the third element of A, for example, you could type in A(3) into the calculator. In ALG mode, simply type A(3). In RPN mode, type 'A(3)' ENTER EVAL .

You can operate with elements of the array by writing and evaluating algebraic expressions such as:



```
:A(2)+A(5)
                  -7
:A(1)-A(4)
                   3
:A(3)·A(2)
                   6
```

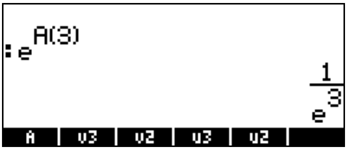


: $\frac{A(3)}{A(5)}$

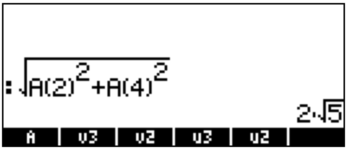
$\frac{3}{5}$

:LN(A(5))

LN(5)+i·π

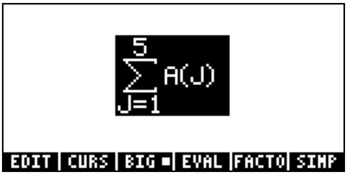


```
:e^A(3)
                    1
                   ───
                    3
                   e
```



```
:√(A(2)²+A(4)²)
                   2·√5
```

More complicated <u>*expressions involving elements*</u> of A can also be written. For example, using the Equation Writer (↱ EQW ), we can write the following summation of the elements of A:

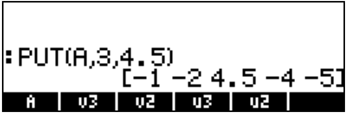


$$\sum_{J=1}^{5} A(J)$$

Highlighting the entire expression and using the EVAL soft menu key, we get the result: −15.

**Note:** The vector A can also be referred to as an *indexed variable* because the name A represents not one, but many values identified by a sub-index.

To <u>replace an element in an array</u> use function PUT (you can find it in the function catalog ↱ CAT , or in the PRG/LIST/ELEMENTS sub-menu – the later was introduced in Chapter 8). In ALG mode, you need to use function PUT with the following arguments: PUT(*array, location to be replaced, new value*). For example, to change the contents of A(3) to 4.5, use:



```
:PUT(A,3,4.5)
            [-1 -2 4.5 -4 -5]
```

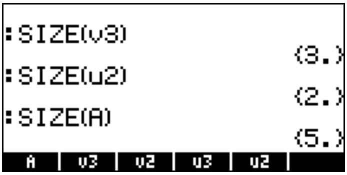
In RPN mode, you can <u>*change the value of an element*</u> of A, by storing a new value in that particular element. For example, if we want to change the contents of A(3) to read 4.5 instead of its current value of –3., use:

4 • 5 ENTER ' ALPHA A ↰ () 3 ENTER STO▶

To verify that the change took place use: ↱ A . The result now shown is: [-1 -2 4.5 -4 -5 ].

**Note**: This approach for changing the value of an array element is not allowed in ALG mode, if you try to store 4.5 into A(3) in this mode you get the following error message: Invalid Syntax.

To find the length of a vector you can use the function SIZE, available through the command catalog (N) or through the PRG/LIST/ELEMENTS sub-menu. Some examples, based on the arrays or vectors stored previously, are shown below:



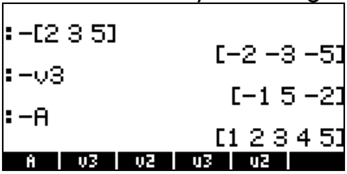
```
:SIZE(v3)
                 {3.}
:SIZE(u2)
                 {2.}
:SIZE(A)
                 {5.}
```

# Simple operations with vectors

To illustrate operations with vectors we will use the vectors A, u2, u3, v2, and v3, stored in an earlier exercise.

## Changing sign

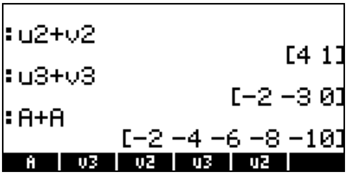
To change the sign of a vector use the key +/-, e.g.,



```
:-[2 3 5]
           [-2 -3 -5]
:-v3
            [-1 5 -2]
:-A
          [1 2 3 4 5]
```

## Addition, subtraction

Addition and subtraction of vectors require that the two vector operands have the same length:

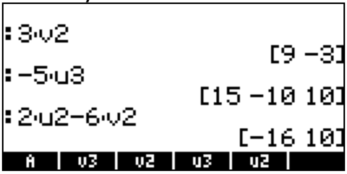


```
:u2+v2
                [4 1]
:u3+v3
           [-2 -3 0]
:A+A
  [-2 -4 -6 -8 -10]
```

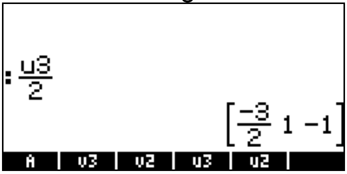
Attempting to add or subtract vectors of different length produces an error message (Invalid Dimension), e.g., v2+v3, u2+u3, A+v3, etc.

## Multiplication by a scalar, and division by a scalar

Multiplication by a scalar or division by a scalar is straightforward:



```
:3·v2
                [9 -3]
:-5·u3
          [15 -10 10]
:2·u2-6·v2
            [-16 10]
```

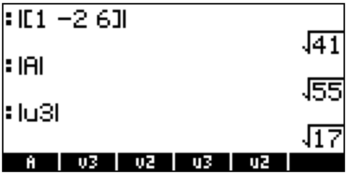


$: \frac{u3}{2}$

$\left[\frac{-3}{2}\ 1\ -1\right]$

## Absolute value function

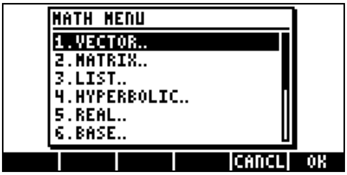
The absolute value function (ABS), when applied to a vector, produces the magnitude of the vector. For a vector A = [$A_1$,$A_2$,…,$A_n$], the magnitude is defined as $|A| = \sqrt{A_x^2 + A_y^2 + \cdots + A_z^2}$. In the ALG mode, enter the function name followed by the vector argument. For example: `ABS([1,-2,6])`, `ABS(A)`, `ABS(u3)`, will show in the screen as follows:



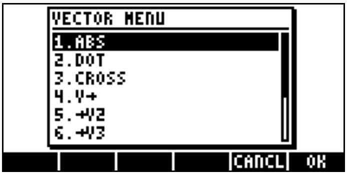
```
:|[1 -2 6]|
                 √41
:|A|
                 √55
:|u3|
                 √17
```

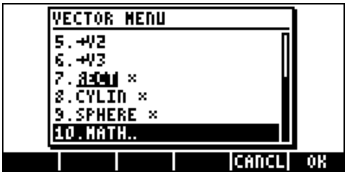
# The MTH/VECTOR menu

The MTH menu (⬅ MTH ) contains a menu of functions that specifically to vector objects:



The VECTOR menu contains the following functions (system flag 117 set to CHOOSE boxes):



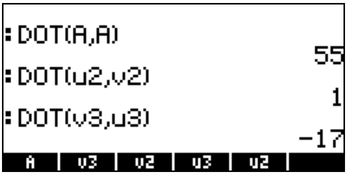


## Magnitude

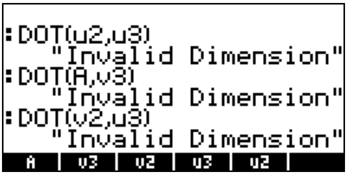
The magnitude of a vector, as discussed earlier, can be found with function ABS. This function is also available from the keyboard (↰ ABS ). Examples of application of function ABS were shown above.

## Dot product

Function DOT is used to calculate the dot product of two vectors of the same length. Some examples of application of function DOT, using the vectors A, u2, u3, v2, and v3, stored earlier, are shown next in ALG mode. Attempts to calculate the dot product of two vectors of different length produce an error message:



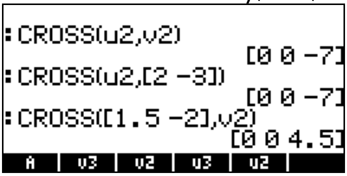
```
:DOT(A,A)
                     55
:DOT(u2,v2)
                      1
:DOT(v3,u3)
                    -17
```



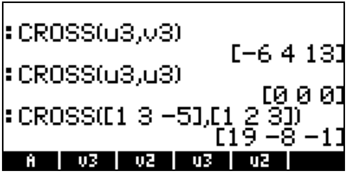
```
:DOT(u2,u3)
   "Invalid Dimension"
:DOT(A,v3)
   "Invalid Dimension"
:DOT(v2,u3)
   "Invalid Dimension"
```

## Cross product

Function CROSS is used to calculate the cross product of two 2-D vectors, of two 3-D vectors, or of one 2-D and one 3-D vector. For the purpose of calculating a cross product, a 2-D vector of the form [$A_x$, $A_y$], is treated as the 3-D vector [$A_x$, $A_y$,0]. Examples in ALG mode are shown next for two 2-D and two 3-D vectors. Notice that the cross product of two 2-D vectors will produce a vector in the z-direction only, i.e., a vector of the form [0, 0, $C_z$]:

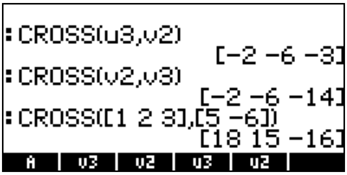


```
:CROSS(u2,v2)
                [0 0 -7]
:CROSS(u2,[2 -3])
                [0 0 -7]
:CROSS([1.5 -2],v2)
              [0 0 4.5]
```



```
:CROSS(u3,v3)
              [-6 4 13]
:CROSS(u3,u3)
                [0 0 0]
:CROSS([1 3 -5],[1 2 3])
              [19 -8 -1]
```

Examples of cross products of one 3-D vector with one 2-D vector, or vice versa, are presented next:

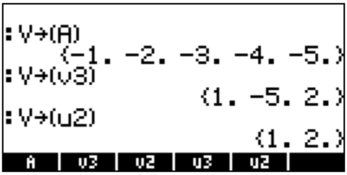


```
:CROSS(u3,v2)
                [-2 -6 -3]
:CROSS(v2,v3)
               [-2 -6 -14]
:CROSS([1 2 3],[5 -6])
               [18 15 -16]
```

Attempts to calculate a cross product of vectors of length other than 2 or 3, produce an error message (Invalid Dimension), e.g., CROSS(v3,A), etc.

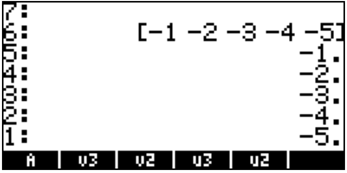
## Decomposing a vector

Function V→ is used to decompose a vector into its elements or components. If used in the ALG mode, V→ will provide the elements of the vector in a list, e.g.,



```
:V→(A)
    {-1. -2. -3. -4. -5.}
:V→(v3)
             {1. -5. 2.}
:V→(u2)
                 {1. 2.}
```

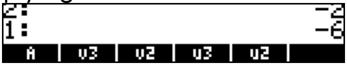
In the RPN mode, application of function V→ will list the components of a vector in the stack, e.g., V→(A) will produce the following output in the RPN stack (vector A is listed in stack level 6:).



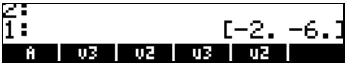
```
7:
6:        [-1 -2 -3 -4 -5]
5:                     -1.
4:                     -2.
3:                     -3.
2:                     -4.
1:                     -5.
```

## Building a two-dimensional vector

Function →V2 is used in the RPN mode to build a vector with the values in stack levels 1: and 2:. The following screen shots show the stack before and after applying function →V2:



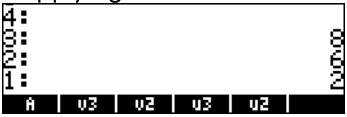
```
2:                      -2
1:                      -6
```



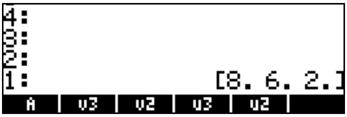
```
2:
1:               [-2. -6.]
```

## Building a three-dimensional vector

Function →V3 is used in the RPN mode to build a vector with the values in stack levels 1: , 2:, and 3:. The following screen shots show the stack before and after applying function →V2:



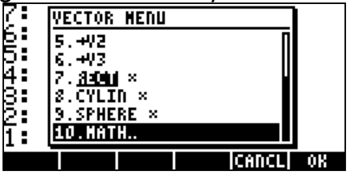
```
4:
3:                    8
2:                    6
1:                    2
 A | v3 | v2 | v3 | v2 |
```



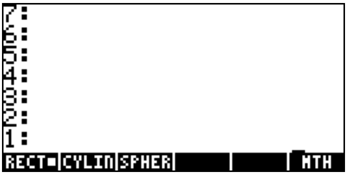
```
4:
3:
2:
1:           [8. 6. 2.]
 A | v3 | v2 | v3 | v2 |
```

## Changing coordinate system

Functions RECT, CYLIN, and SPHERE are used to change the current coordinate system to rectangular (Cartesian), cylindrical (polar), or spherical coordinates. The current system is shown highlighted in the corresponding CHOOSE box (system flag 117 unset), or selected in the corresponding SOFT menu label (system flag 117 set). In the following figure the RECTangular coordinate system is shown as selected in these two formats:

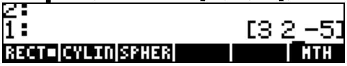


```
VECTOR MENU
5.→V2
6.→V3
7.RECT ▪
8.CYLIN ▪
9.SPHERE ▪
10.MATH..
 |  |  |  |CANCL| OK
```



```
7:
6:
5:
4:
3:
2:
1:
RECT▪|CYLIN|SPHER|  |  | MTH
```

When the rectangular, or Cartesian, coordinate system is selected, the top line of the display will show an XYZ field, and any 2-D or 3-D vector entered in the calculator is reproduced as the (x,y,z) components of the vector. Thus, to enter the vector A = 3**i**+2**j**-5**k**, we use [3,2,-5], and the vector is shown as:

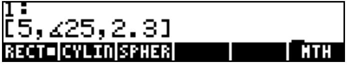


```
2:
1:            [3 2 -5]
RECT▪|CYLIN|SPHER|  |  | MTH
```

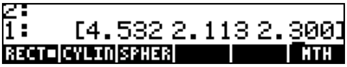
If instead of entering Cartesian components of a vector we enter cylindrical (polar) components, we need to provide the magnitude, r, of the projection of the vector on the x-y plane, an angle θ (in the current angular measure) representing the inclination of r with respect to the positive x-axis , and a z-component of the vector. The angle θ must be entered preceded by the angle character (∠), generated by using ALPHA ↱ 6 . For example, suppose that we have a vector with r = 5, θ = 25° (DEG should be selected as the angular measure), and z = 2.3, we can enter this vector in the following way:

↰ [] 5 ↱ ⎵, ALPHA ↱ 6 2 5 ↱ ⎵, 2 . 3

Before pressing ENTER, the screen will look as in the left-hand side of the following figure. After pressing ENTER, the screen will look as in the right-hand side of the figure (For this example, the numerical format was changed to Fix, with three decimals).



```
1:
[5,∡25,2.3]
```

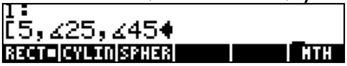


```
2:
1:    [4.532 2.113 2.300]
```

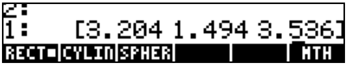
Notice that the vector is displayed in Cartesian coordinates , with components x = r cos(θ), y = r sin(θ), z = z, even though we entered it in polar coordinates. This is because the vector display will default to the current coordinate system. For this case, we have x = 4.532, y = 2.112, and z = 2.300.

Suppose that we now enter a vector in spherical coordinates (i.e., in the form (ρ,θ,ϕ), where ρ is the length of the vector, θ is the angle that the xy projection of the vector forms with the positive side of the x-axis, and ϕ is the angle that ρ forms with the positive side of the z axis), with ρ = 5, θ = 25°, and ϕ = 45°. We will use: ↰ [] 5 ↱ ⎵, ALPHA ↱ 6 2 5 ⎵, ALPHA ↱ 6 4 5

The figure below shows the transformation of the vector from spherical to Cartesian coordinates, with x = ρ sin(ϕ) cos(θ), y = ρ sin (ϕ) cos (θ), z = ρ cos(ϕ). For this case, x = 3.204, y = 1.494, and z = 3.536.

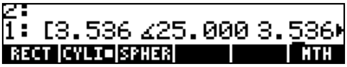


```
1:
[5,∡25,∡45◆
```



```
2:
1:    [3.204 1.494 3.536]
```

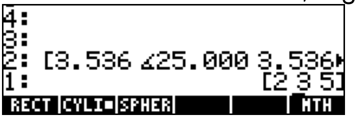
If the CYLINdrical system is selected, the top line of the display will show an R∠Z field, and a vector entered in cylindrical coordinates will be shown in its cylindrical (or polar) coordinate form (r,θ,z). To see this in action, change the coordinate system to CYLINdrical and watch how the vector displayed in the last screen changes to its cylindrical (polar) coordinate form. The second component is shown with the angle character in front to emphasize its angular nature.



```
2:
1: [3.536 ∡25.000 3.536▸
```

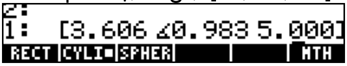
The conversion from Cartesian to cylindrical coordinates is such that $r = (x^2+y^2)^{1/2}$, $\theta = \tan^{-1}(y/x)$, and $z = z$. For the case shown above the transformation was such that $(x,y,z) = (3.204, 2.112, 2.300)$, produced $(r,\theta,z) = (3.536,25°,3.536)$.

At this point, change the angular measure to Radians. If we now enter a vector of integers in Cartesian form, even if the CYLINdrical coordinate system is active, it will be shown in Cartesian coordinates, e.g.,



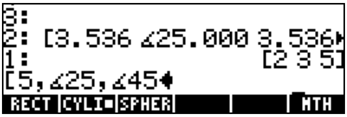
```
4:
3:
2: [3.536 ∡25.000 3.536▸
1:                [2 3 5]
```

This is because the integer numbers are intended for use with the CAS and, therefore, the components of this vector are kept in Cartesian form. To force the conversion to polar coordinates enter the vector components as real numbers (i.e., add a decimal point), e.g., [2., 3., 5.].

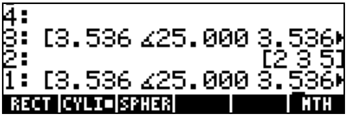


```
2:
1:  [3.606 ∡0.983 5.000]
```

With the cylindrical coordinate system selected, if we enter a vector in spherical coordinates it will be automatically transformed to its cylindrical (polar) equivalent $(r,\theta,z)$ with $r = \rho \sin \phi$, $\theta = \theta$, $z = \rho \cos \phi$. For example, the following figure shows the vector entered in spherical coordinates, and transformed to polar coordinates. For this case, $\rho = 5$, $\theta = 25°$, and $\phi = 45°$, while the transformation shows that $r = 3.563$, and $z = 3.536$.

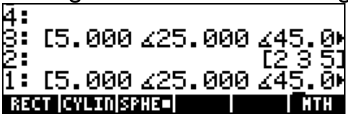


```
3:
2: [3.536 ∡25.000 3.536▸
1:                [2 3 5]
[5,∡25,∡45◆
```



```
4:
3: [3.536 ∡25.000 3.536▸
2:                [2 3 5]
1: [3.536 ∡25.000 3.536▸
```

Next, let's change the coordinate system to spherical coordinates by using function SPHERE from the VECTOR sub-menu in the MTH menu. When this coordinate system is selected, the display will show the R∠∠ format in the top line. The last screen will change to show the following:



```
4:
3: [5.000 ∡25.000 ∡45.0▸
2:                [2 3 5]
1: [5.000 ∡25.000 ∡45.0▸
```

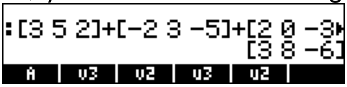
Notice that the vectors that were written in cylindrical polar coordinates have now been changed to the spherical coordinate system. The transformation is such that $\rho = (r^2+z^2)^{1/2}$, $\theta = \theta$, and $\phi = \tan^{-1}(r/z)$. However, the vector that originally was set to Cartesian coordinates remains in that form.

# Application of vector operations

This section contains some examples of vector operations that you may encounter in Physics or Mechanics applications.

## Resultant of forces

Suppose that a particle is subject to the following forces (in N): $\mathbf{F}_1$ = 3**i**+5**j**+2**k**, $\mathbf{F}_2$ = -2**i**+3**j**-5**k**, and $F_3$ = 2**i**-3**k**. To determine the resultant, i.e., the sum, of all these forces, you can use the following approach in ALG mode:



```
:[3 5 2]+[-2 3 -5]+[2 0 -3▸
                   [3 8 -6]
```

Thus, the resultant is R = $\mathbf{F}_1$+ $\mathbf{F}_2$ + $\mathbf{F}_3$ = (3**i**+8**j**-6**k**)N. RPN mode use:

```
[3,5,2] ENTER [-2,3,-5] ENTER [2,0,3] ENTER + +
```

## Angle between vectors

The angle between two vectors **A**, **B**, can be found as $\theta = \cos^{-1}(\mathbf{A}\bullet\mathbf{B}/|\mathbf{A}||\mathbf{B}|)$

Suppose that you want to find the angle between vectors **A** = 3**i**-5**j**+6**k**, **B** = 2**i**+**j**-3**k**, you could try the following operation (angular measure set to degrees) in ALG mode:

1 - Enter vectors [3,-5,6], press ENTER, [2,1,-3], press ENTER.

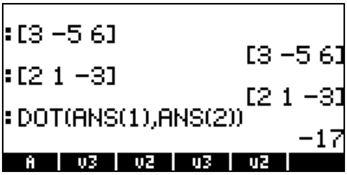
2 - DOT(ANS(1),ANS(2)) calculates the dot product

3 - ABS(ANS(3))*ABS((ANS(2)) calculates product of magnitudes

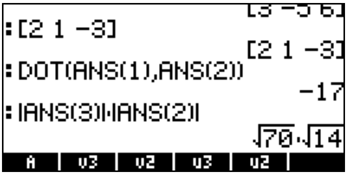
4 - ANS(2)/ANS(1) calculates cos(θ)

5 – ACOS(ANS(1)), followed by ,→NUM(ANS(1)), calculates θ

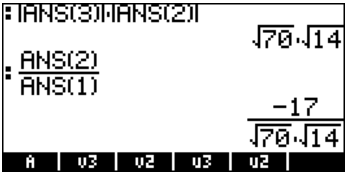
The steps are shown in the following screens (ALG mode, of course):



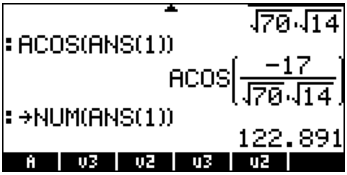
```
:[3 -5 6]
                    [3 -5 6]
:[2 1 -3]
                    [2 1 -3]
:DOT(ANS(1),ANS(2))
                         -17
```



```
                    [3 -5 6]
:[2 1 -3]
                    [2 1 -3]
:DOT(ANS(1),ANS(2))
                         -17
:|ANS(3)|·|ANS(2)|
                     √70·√14
```



```
:|ANS(3)|·|ANS(2)|
                    √70·√14
: ANS(2)
  ANS(1)
                        -17
                    √70·√14
```



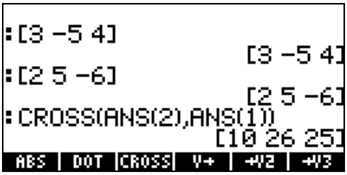
```
                    √70·√14
:ACOS(ANS(1))
                      ⎛  -17  ⎞
                  ACOS⎜───────⎟
                      ⎝√70·√14⎠
:→NUM(ANS(1))
                    122.891
```

Thus, the result is θ = 122.891°. In RPN mode use the following:

[3,-5,6] ENTER [2,1,-3] ENTER DOT
[3,-5,6] ENTER ABS [2,1,-3] ENTER ABS ×
÷ ACOS →NUM

## Moment of a force

The moment exerted by a force **F** about a point O is defined as the cross-product **M** = **r**×**F**, where **r**, also known as the arm of the force, is the position vector based at O and pointing towards the point of application of the force. Suppose that a force **F** = (2**i**+5**j**-6**k**) N has an arm **r** = (3**i**-5**j**+4**k**)m. To determine the moment exerted by the force with that arm, we use function CROSS as shown next:



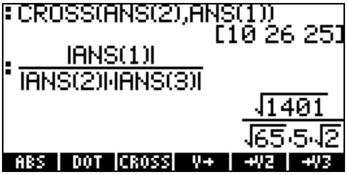
```
:[3 -5 4]
                  [3 -5 4]
:[2 5 -6]
                  [2 5 -6]
:CROSS(ANS(2),ANS(1))
                [10 26 25]
```

Thus, **M** = (10**i**+26**j**+25**k**) m·N. We know that the magnitude of **M** is such that |**M**| = |**r**||**F**|sin(θ), where θ is the angle between **r** and **F**. We can find this angle as, θ = $\sin^{-1}$(|**M**| /|**r**||**F**|) by the following operations:

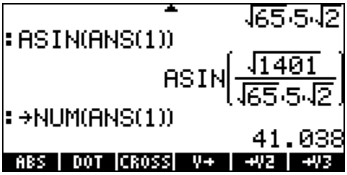
1 – ABS(ANS(1))/(ABS(ANS(2))*ABS(ANS(3)) calculates sin(θ)

2 – ASIN(ANS(1)), followed by →NUM(ANS(1)) calculates θ

These operations are shown, in ALG mode, in the following screens:



```
:CROSS(ANS(2),ANS(1))
                [10 26 25]
:      |ANS(1)|
  |ANS(2)|·|ANS(3)|
                     √1401
                   √65·5·√2
```



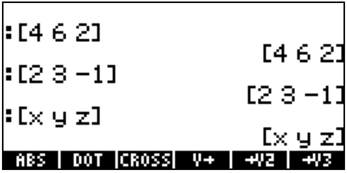
```
                   √65·5·√2
:ASIN(ANS(1))
                  ⎛  √1401  ⎞
              ASIN⎜─────────⎟
                  ⎝√65·5·√2 ⎠
:→NUM(ANS(1))
                     41.038
```

Thus the angle between vectors **r** and **F** is θ = 41.038°. RPN mode, we can use: `3,-5,4]` ENTER `[2,5,-6]` ENTER `CROSS` `ABS` `[3,-5,4]` ENTER `ABS` `[2,5,-6]` ENTER `ABS` × ÷ `ASIN` `→NUM`

## Equation of a plane in space

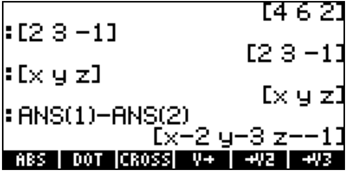
Given a point in space $P_0(x_0,y_0,z_0)$ and a vector $\mathbf{N} = N_x\mathbf{i}+N_y\mathbf{j}+N_z\mathbf{k}$ normal to a plane containing point $P_0$, the problem is to find the equation of the plane. We can form a vector starting at point $P_0$ and ending at point P(x,y,z), a generic point in the plane. Thus, this vector $\mathbf{r} = P_0P = (x\text{-}x_0)\mathbf{i}+ (y\text{-}y_0)\mathbf{j} + (z\text{-}z_0)\mathbf{k}$, is perpendicular to the normal vector **N**, since **r** is contained entirely in the plane. We learned that for two normal vectors **N** and **r**, **N•r** =0. Thus, we can use this result to determine the equation of the plane.

To illustrate the use of this approach, consider the point $P_0(2,3,-1)$ and the normal vector $\mathbf{N} = 4\mathbf{i}+6\mathbf{j}+2\mathbf{k}$, we can enter vector **N** and point $P_0$ as two vectors, as shown below. We also enter the vector [x,y,z] last:



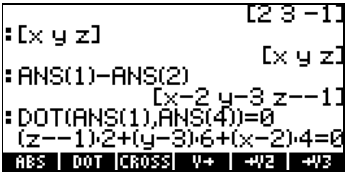
```
:[4 6 2]
                [4 6 2]
:[2 3 -1]
               [2 3 -1]
:[x y z]
                [x y z]
```

Next, we calculate vector $P_0P$ = **r** as ANS(1) – ANS(2), i.e.,



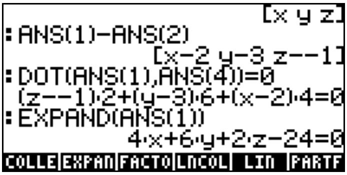
```
                [4 6 2]
:[2 3 -1]
               [2 3 -1]
:[x y z]
                [x y z]
:ANS(1)-ANS(2)
         [x-2 y-3 z--1]
```

Finally, we take the dot product of ANS(1) and ANS(4) and make it equal to zero to complete the operation **N•r** =0:



```
               [2 3 -1]
:[x y z]
                [x y z]
:ANS(1)-ANS(2)
         [x-2 y-3 z--1]
:DOT(ANS(1),ANS(4))=0
 (z--1)·2+(y-3)·6+(x-2)·4=0
```

We can now use function EXPAND (in the ALG menu) to expand this expression:



```
                    [x y z]
:ANS(1)-ANS(2)
             [x-2 y-3 z--1]
:DOT(ANS(1),ANS(4))=0
 (z--1)·2+(y-3)·6+(x-2)·4=0
:EXPAND(ANS(1))
           4·x+6·y+2·z-24=0
```

Thus, the equation of the plane through point $P_0(2,3,-1)$ and having normal vector **N** = 4**i**+6**j**+2**k**, is 4x + 6y + 2z – 24 = 0. In RPN mode, use:

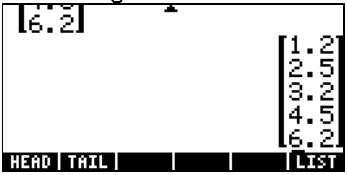
`[2,3,-1]` ENTER `['x','y','z']` ENTER − `[4,6,2] DOT EXPAND`

## Row vectors, column vectors, and lists

The vectors presented in this chapter are all row vectors. In some instances, it is necessary to create a column vector (e.g., to use the pre-defined statistical functions in the calculator). The simplest way to enter a column vector is by enclosing each vector element within brackets, all contained within an external set of brackets. For example, enter:

`[[1.2],[2.5],[3.2],[4.5],[6.2]]` ENTER

This is represented as the following column vector:



```
[6.2]
                 [1.2]
                 [2.5]
                 [3.2]
                 [4.5]
                 [6.2]
```

In this section we will showing you ways to transform: a column vector into a row vector, a row vector into a column vector, a list into a vector, and a vector (or matrix) into a list.

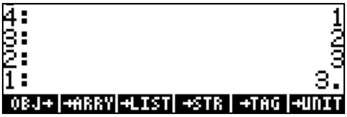
We first demonstrate these transformations using the RPN mode. In this mode, we will use functions OBJ→, →LIST, →ARRY and DROP to perform the transformation. To facilitate accessing these functions we will set system flag 117 to SOFT menus (see Chapter 1). With this flag set, functions OBJ→, →ARRY, and →LIST will be accessible by using ← PRG TYPE. Functions

OBJ→, →ARRY, and →LIST will be available in soft menu keys F1 , F2 , and F3 . Function DROP is available by using ← PRG STACK DROP.

Following we introduce the operation of functions OBJ→, →LIST, →ARRY, and DROP with some examples.

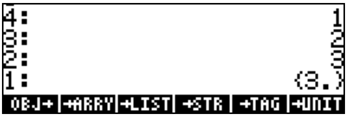
## Function OBJ→

This function decomposes an object into its components. If the argument is a list, function OBJ→ will list the list elements in the stack, with the number of elements in stack level 1, for example: {1,2,3} ENTER ← PRG TYPE OBJ→ results in:



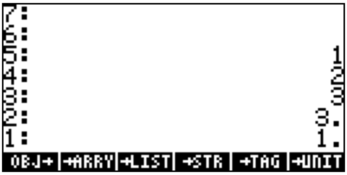
```
4:                 1
3:                 2
2:                 3
1:                3.
OBJ→ →ARRY →LIST →STR →TAG →UNIT
```

When function OBJ→ is applied to a vector, it will list the elements of the vector in the stack, with the number of elements in level 1: enclosed in braces (a list). The following example illustrates this application: [1,2,3] ENTER ← PRG TYPE OBJ→ results in:



```
4:                 1
3:                 2
2:                 3
1:              {3.}
OBJ→ →ARRY →LIST →STR →TAG →UNIT
```

If we now apply function OBJ→ once more, the list in stack level 1:, {3.}, will be decomposed as follows:



```
7:
6:
5:                 1
4:                 2
3:                 3
2:                3.
1:                1.
OBJ→ →ARRY →LIST →STR →TAG →UNIT
```

## Function →LIST

This function is used to create a list given the elements of the list and the list length or size. In RPN mode, the list size, say, n, should be placed in stack level 1:. The elements of the list should be located in stack levels 2:, 3:, …, n+1:. For example, to create the list {1, 2, 3}, type: 1 ENTER 2 ENTER 3 ENTER 3 ENTER ↰ PRG TYPE →LIST.

## Function →ARRY

This function is used to create a vector or a matrix. In this section, we will use it to build a vector or a column vector (i.e., a matrix of n rows and 1 column). To build a regular vector we enter the elements of the vector in the stack, and in stack level 1: we enter the vector size as a list, e.g., 1 ENTER 2 ENTER 3 ENTER ↰ {} 3 ENTER ↰ PRG TYPE →ARRY.

To build a column vector of n elements, enter the elements of the vector in the stack, and in stack level 1 enter the list {n 1}. For example, 1 ENTER 2 ENTER 3 ENTER ↰ {} 1 ↱ , 3 ENTER ↰ PRG TYPE →ARRY.

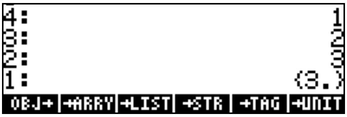
## Function DROP

This function has the same effect as the delete key (◀).

## Transforming a row vector into a column vector

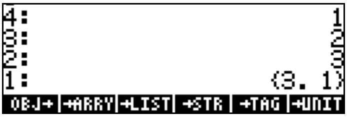
We illustrate the transformation with vector `[1,2,3]`. Enter this vector into the RPN stack to follow the exercise. To transform a row vector into a column vector, we need to carry on the following operations in the RPN stack:

1 - Decompose the vector with function OBJ→



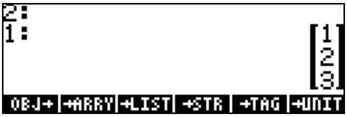
```
4:                    1
3:                    2
2:                    3
1:                  {3.}
```

2 - Press 1 + to transform the list in stack level 1: from {3} to {3,1}



```
4:                    1
3:                    2
2:                    3
1:               {3. 1}
```

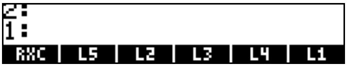
3 - Use function →ARRY to build the column vector



```
2:
1:                  [1]
                    [2]
                    [3]
```

These three steps can be put together into a UserRPL program, entered as follows (in RPN mode, still): ↱ « » ↰ PRG TYPE OBJ→ 1 + →ARRY ENTER ' ALPHA ALPHA R X C ENTER STO▸

A new variable, RXC, will be available in the soft menu labels after pressing VAR :



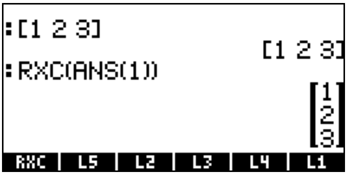
```
2:
1:
```

Press ↱ RXC to see the program contained in the variable RXC:

<< OBJ→ 1 + →ARRY >>

This variable, RXC, can now be used to directly transform a row vector to a column vector. In RPN mode, enter the row vector, and then press RXC. Try, for example: [1,2,3] ENTER RXC.

After having defined this variable , we can use it in ALG mode to transform a row vector into a column vector. Thus, change your calculator's mode to ALG and try the following procedure: [1,2,3] ENTER VAR RXC ↰ () ↰ ANS , resulting in:

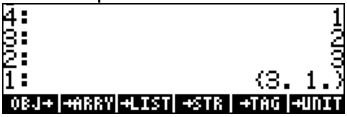


```
:[1 2 3]
                    [1 2 3]
:RXC(ANS(1))
                        [1]
                        [2]
                        [3]
```

## Transforming a column vector into a row vector

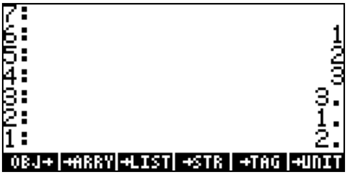
To illustrate this transformation, we'll enter the column vector [[1],[2],[3]] in RPN mode. Then, follow the next exercise to transform a row vector into a column vector:

1 - Use function OBJ→ to decompose the column vector



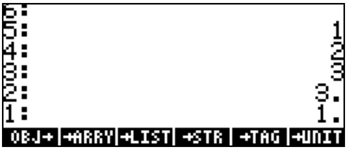
```
4:                        1
3:                        2
2:                        3
1:                  {3. 1.}
```

2 - Use function OBJ→ to decompose the list in stack level 1:



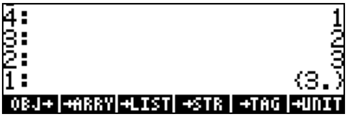
```
7:
6:                        1
5:                        2
4:                        3
3:                       3.
2:                       1.
1:                       2.
```

3 - Press the delete key ◄ (also known as function DROP) to eliminate the number in stack level 1:



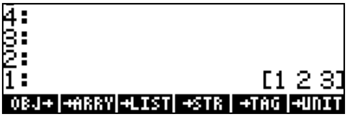
```
6:
5:                 1
4:                 2
3:                 3
2:                3.
1:                1.
OBJ→ →ARRY →LIST →STR →TAG →UNIT
```

4 - Use function →LIST to create a list



```
4:                 1
3:                 2
2:                 3
1:              {3.}
OBJ→ →ARRY →LIST →STR →TAG →UNIT
```

5 - Use function →ARRY to create the row vector

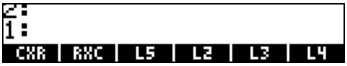


```
4:
3:
2:
1:           [1 2 3]
OBJ→ →ARRY →LIST →STR →TAG →UNIT
```

These five steps can be put together into a UserRPL program, entered as follows (in RPN mode, still):

→ « » ← PRG TYPE OBJ→ OBJ→
← PRG STACK DROP ← PRG TYPE →LIST →ARRY ENTER
' ALPHA ALPHA C X R ENTER STO▶

A new variable, CXR, will be available in the soft menu labels after pressing VAR:



```
2:
1:
CXR  RXC  L5  L2  L3  L4
```

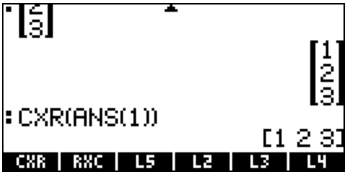
Press → CXR to see the program contained in the variable CXR:

```
<< OBJ→ OBJ→ DROP →ARRY >>
```

This variable, CXR, can now be used to directly transform a column vector to a row vector. In RPN mode, enter the column vector, and then press CXR. Try, for example: `[[1],[2],[3]]` ENTER CXR.

After having defined variable CXR, we can use it in ALG mode to transform a row vector into a column vector. Thus, change your calculator's mode to ALG and try the following procedure:

`[[1],[2],[3]]` ENTER VAR CXR ← () ← ANS resulting in:

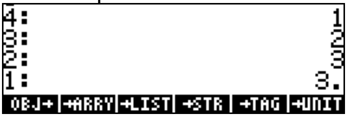


```
:[2
  3]
                          [1
                           2
                           3]
:CXR(ANS(1))
                     [1 2 3]
```

## Transforming a list into a vector

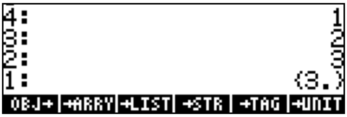
To illustrate this transformation, we'll enter the list {1,2,3} in RPN mode. Then, follow the next exercise to transform a list into a vector:

1 - Use function OBJ→ to decompose the column vector



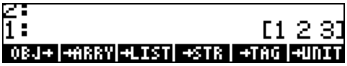
```
4:                          1
3:                          2
2:                          3
1:                         3.
```

2 - Type a 1 and use function →LIST to create a list in stack level 1:



```
4:                          1
3:                          2
2:                          3
1:                       {3.}
```

3 - Use function →ARRY to create the vector

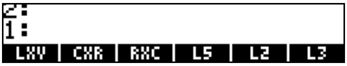


```
2:
1:                    [1 2 3]
```

These three steps can be put together into a UserRPL program, entered as follows (in RPN mode):

↱ « » ↰ PRG TYPE OBJ→ 1 →LIST →ARRY ENTER
' ALPHA ALPHA L X V ENTER STO▶

A new variable, LXV, will be available in the soft menu labels after pressing VAR :



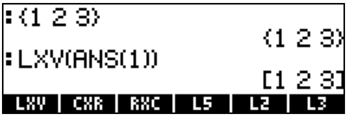
```
2:
1:
```

Press ↱ LXV to see the program contained in the variable LXV:

```
<< OBJ→ 1 →LIST →ARRY >>
```

This variable, LXV, can now be used to directly transform a list into a vector. In RPN mode, enter the list, and then press LXV. Try, for example: {1,2,3} ENTER LXV.

After having defined variable LXV, we can use it in ALG mode to transform a list into a vector. Thus, change your calculator's mode to ALG and try the following procedure: {1,2,3} ENTER VAR LXV ← () ← ANS , resulting in:



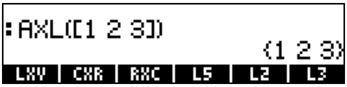
```
:{1 2 3}
                    {1 2 3}
:LXV(ANS(1))
                    [1 2 3]
```

## Transforming a vector (or matrix) into a list

To transform a vector into a list, the calculator provides function AXL. You can find this function through the command catalog, as follows:

↱ CAT ALPHA ALPHA A X L ALPHA OK

As an example, apply function AXL to the vector [1,2,3] in RPN mode by using:[1,2,3] ENTER AXL. The following screen shot shows the application of function AXL to the same vector in ALG mode.



```
:AXL([1 2 3])
                    {1 2 3}
```

# Chapter 10

# Creating and manipulating matrices

This chapter shows a number of examples aimed at creating matrices in the calculator and demonstrating manipulation of matrix elements.

## Definitions

A matrix is simply a rectangular array of objects (e.g., numbers, algebraics) having a number of rows and columns. A matrix **A** having n rows and m columns will have, therefore, n×m elements. A generic element of the matrix is represented by the indexed variable $a_{ij}$, corresponding to row i and column j. With this notation we can write matrix **A** as **A** = $[a_{ij}]_{n\times m}$ . The full matrix is shown next:

$$\mathbf{A} = [a_{ij}]_{n\times m} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}.$$

A matrix is square if m = n. The transpose of a matrix is constructed by swapping rows for columns and vice versa. Thus, the transpose of matrix **A**, is $\mathbf{A}^T = [(a^T)_{ij}]_{m\times n} = [a_{ji}]_{m\times n}$. The main diagonal of a square matrix is the collection of elements $a_{ii}$. An identity matrix, $\mathbf{I}_{n\times n}$, is a square matrix whose main diagonal elements are all equal to 1, and all off-diagonal elements are zero. For example, a 3×3 identity matrix is written as

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

An identity matrix can be written as $\mathbf{I}_{n\times n} = [\delta_{ij}]$, where $\delta_{ij}$ is a function known as Kronecker's delta, and defined as

$$\delta_{ij} = \begin{cases} 1, & if\ i = j \\ 0, & if\ i \neq j \end{cases}.$$

# Entering matrices in the stack

In this section we present two different methods to enter matrices in the calculator stack: (1) using the Matrix Editor, and (2) typing the matrix directly into the stack.

## Using the matrix editor

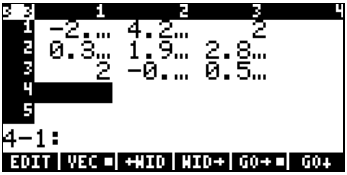
As with the case of vectors, discussed in Chapter 9, matrices can be entered into the stack by using the matrix editor. For example, to enter the matrix:

$$\begin{bmatrix} -2.5 & 4.2 & 2.0 \\ 0.3 & 1.9 & 2.8 \\ 2 & -0.1 & 0.5 \end{bmatrix},$$

first, start the matrix writer by using ← MTRW . Make sure that the option GO→■ is selected. Then use the following keystrokes:

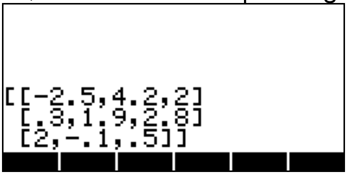
2 . 5 +/- ENTER 4 . 2 ENTER 2 ENTER ▼ ◀ ◀ ◀
. 3 ENTER 1 . 9 ENTER 2 . 8 ENTER
2 ENTER . 1 +/- ENTER . 5 ENTER

At this point, the Matrix Writer screen may look like this:

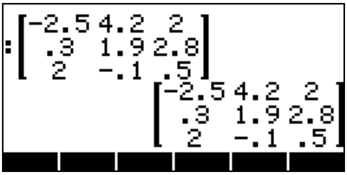


```
3 3      1       2       3    4
1   -2.…  4.2…        2
2   0.3…  1.9…  2.8…
3      2 -0.…  0.5…
4
5
4-1:
```

Press ENTER once more to place the matrix on the stack. The ALG mode stack is shown next, before and after pressing , once more:

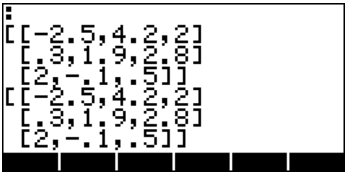


```
[[-2.5,4.2,2]
 [.3,1.9,2.8]
 [2,-.1,.5]]
```



```
:⎡-2.5 4.2  2 ⎤
 ⎢  .3  1.9 2.8⎥
 ⎣  2   -.1  .5⎦
           ⎡-2.5 4.2  2 ⎤
           ⎢  .3  1.9 2.8⎥
           ⎣  2   -.1  .5⎦
```

If you have selected the textbook display option (using MODE DISP and checking off ✓Textbook), the matrix will look like the one shown above. Otherwise, the display will show:



```
:
[[-2.5,4.2,2]
 [.3,1.9,2.8]
 [2,-.1,.5]]
[[-2.5,4.2,2]
 [.3,1.9,2.8]
 [2,-.1,.5]]
```

The display in RPN mode will look very similar to these.

**Note**: Details on the use of the matrix writer were presented in Chapter 9.

### Typing in the matrix directly into the stack

The same result as above can be achieved by entering the following directly into the stack:

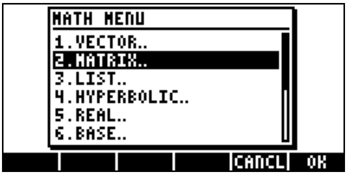
↰ []
↰ [] 2 . 5 +/- ↱ , 4 . 2 ↱ , 2 ▶
↱ ,
↰ [] . 3 ↱ , 1 . 9 ↱ , 2 . 8 ▶
↱ ,
↰ [] 2 ↱ , . 1 +/- ↱ , . 5

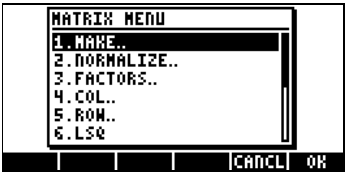
Thus, to enter a matrix directly into the stack open a set of brackets (↰ [] ) and enclose each row of the matrix with an additional set of brackets (↰ [] ). Commas (↱ , .) should separate the elements of each row, as well as the brackets between rows. (**Note**: In RPN mode, you can omit the inner brackets after the first set has been entered, thus, instead of typing, for example, [[1 2 3] [4 5 6] [7 8 9]], type [[1 2 3] 4 5 6 7 8 9].)

For future exercises, let's save this matrix under the name A. In ALG mode use STO▶ ALPHA A . In RPN mode, use ' ALPHA A STO▶ .

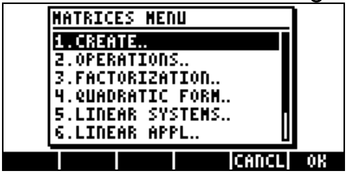
## Creating matrices with calculator functions

Some matrices can be created by using the calculator functions available in either the MTH/MATRIX/MAKE sub-menu within the MTH menu (↰ MTH ),

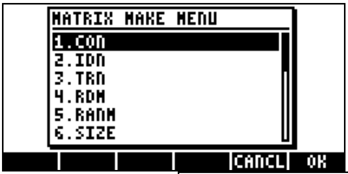


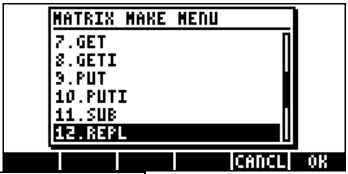


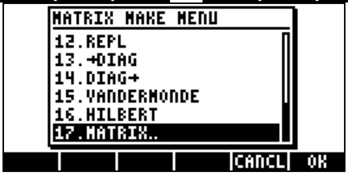
or in the MATRICES/CREATE menu available through ⬅ MATRICES :



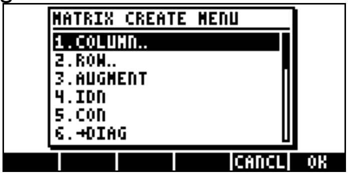
The MTH/MATRIX/MAKE sub menu (let's call it the MAKE menu) contains the following functions:

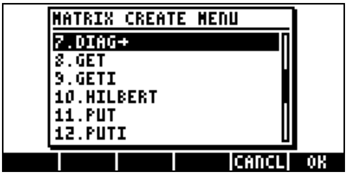


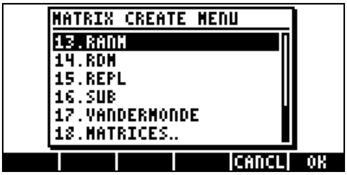




while the MATRICES/CREATE sub-menu (let's call it the CREATE menu) has the following functions:



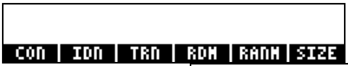


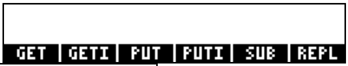


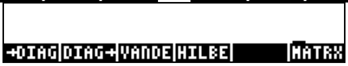
As you can see from exploring these menus (MAKE and CREATE), they both have the same functions GET, GETI, PUT, PUTI, SUB, REPL, RDM, RANM, HILBERT, VANDERMONDE, IDN, CON, →DIAG, and DIAG→. The CREATE menu includes the COLUMN and ROW sub-menus, that are also available under the MTH/MATRIX menu. The MAKE menu includes the functions SIZE, that the CREATE menu does not include. Basically, however, both menus, MAKE and CREATE, provide the user with the same set of functions. In the examples that follow, we will show how to access functions through use of the matrix MAKE menu. At the end of this section we present a table with the keystrokes required to obtain the same functions with the CREATE menu when system flag 117 is set to SOFT menus.

If you have set that system flag (flag 117) to SOFT menu, the MAKE menu will be available through the keystroke sequence: ← MTH MATRX MAKE

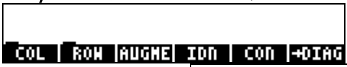
The functions available will be shown as soft-menu key labels as follows (press NXT to move to the next set of functions):

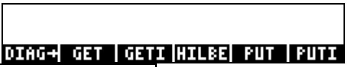


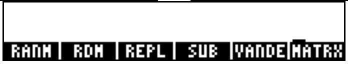




With system flag 117 set to SOFT menus, the functions of the CREATE menu, triggered by ← MATRICES CREAT ,will show as follows:





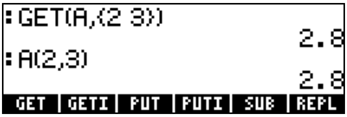


In the next sections we present applications of the matrix functions in the MAKE and CREATE menu.

## Functions GET and PUT

Functions GET, GETI, PUT, and PUTI, operate with matrices in a similar manner as with lists or vectors, i.e., you need to provide the location of the element that you want to GET or PUT. However, while in lists and vectors only one index is required to identify an element, in matrices we need a list of two indices {row, column} to identify matrix elements. Examples of the use of GET and PUT follow.

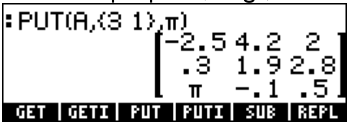
Let's use the matrix we stored above into variable A to demonstrate the use of the GET and PUT functions. For example, to extract element $a_{23}$ from matrix A, in ALG mode, can be performed as follows:



```
:GET(A,{2 3})
                     2.8
:A(2,3)
                     2.8
```

Notice that we achieve the same result by simply typing A(2,3) and pressing ENTER. In RPN mode, this exercise is performed by entering A ENTER 3 ENTER GET, or by using A(2,3) ENTER.

Suppose that we want to place the value 'π' into element $a_{31}$ of the matrix. We can use function PUT for that purpose, e.g.,

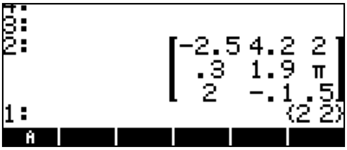


```
:PUT(A,{3 1},π)
              ⎡-2.5 4.2  2 ⎤
              ⎢ .3  1.9 2.8⎥
              ⎣ π   -.1 .5 ⎦
```

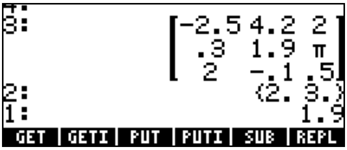
In RPN mode you can use: VAR A {3,1} ENTER ↰ π PUT.
Alternatively, in RPN mode you can use: ↰ π ' A(2,3) ENTER STO▶ .
To see the contents of variable A after this operation, use A.

## Functions GETI and PUTI

Functions PUTI and GETI are used in UserRPL programs since they keep track of an index for repeated application of the PUT and GET functions. The index list in matrices varies by columns first. To illustrate its use, we propose the following exercise in RPN mode: A {2,2}ENTER GETI. Screen shots showing the RPN stack before and after the application of function GETI are shown below:



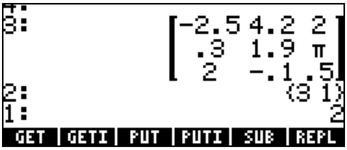
```
3:
2:  [[-2.5 4.2 2 ]
     [ .3  1.9 π ]
     [ 2   -.1 .5]]
1:            {2 2}
```



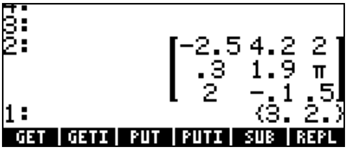
```
3:  [[-2.5 4.2 2 ]
     [ .3  1.9 π ]
     [ 2   -.1 .5]]
2:           {2. 3.}
1:               1.9
```

Notice that the screen is prepared for a subsequent application of GETI or GET, by increasing the column index of the original reference by 1, (i.e., from {2,2} to {2,3}), while showing the extracted value, namely A(2,2) = 1.9, in stack level 1.

Now, suppose that you want to insert the value 2 in element {3 1} using PUTI. Still in RPN mode, try the following keystrokes: ← ←{ 3 1 } ENTER 2 ENTER PUTI. The screen shots below show the RPN stack before and after the application of function PUTI:



```
3:  [[-2.5 4.2 2 ]
     [ .3  1.9 π ]
     [ 2   -.1 .5]]
2:            {3 1}
1:                2
```

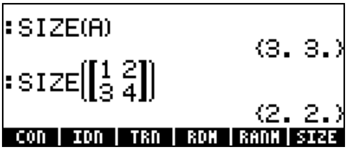


```
3:
2:  [[-2.5 4.2 2 ]
     [ .3  1.9 π ]
     [ 2   -.1 .5]]
1:           {3. 2.}
```

In this case, the 2 was replaced in position {3 1}, i.e., now A(3,1) = 2, and the index list was increased by 1 (by column first), i.e., from {3,1} to {3,2}. The matrix is in level 2, and the incremented index list is in level 1.

## Function SIZE

Function SIZE provides a list showing the number of rows and columns of the matrix in stack level 1. The following screen shows a couple of applications of function SIZE in ALG mode:

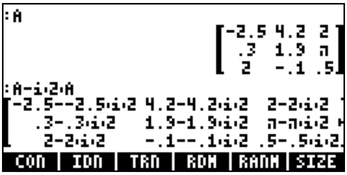


```
:SIZE(A)
                    {3. 3.}
:SIZE([[1 2][3 4]])
                    {2. 2.}
```

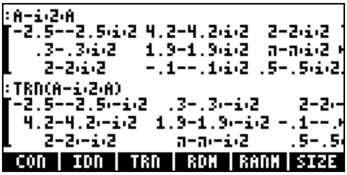
In RPN mode, these exercises are performed by using A SIZE, and [[1,2],[3,4]] ENTER SIZE .

## Function TRN

Function TRN is used to produce the transconjugate of a matrix, i.e., the transpose (TRAN) followed by its complex conjugate (CONJ). For example, the following screen shot shows the original matrix in variable A and its transpose, shown in small font display (see Chapter 1):



```
:A
                      [-2.5 4.2  2 ]
                      [ .3  1.9  π ]
                      [ 2   -.1 .5 ]
:A-i·2·A
[-2.5--2.5·i·2 4.2-4.2·i·2  2-2·i·2
   .3-.3·i·2   1.9-1.9·i·2  π-π·i·2
   2-2·i·2     -.1--.1·i·2 .5-.5·i·2
```

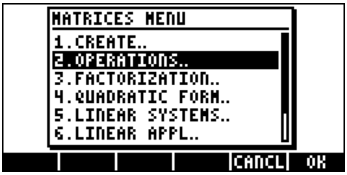


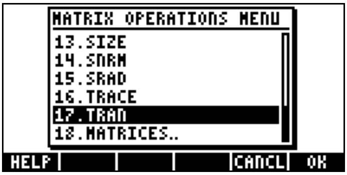
```
:A-i·2·A
[-2.5--2.5·i·2 4.2-4.2·i·2  2-2·i·2
   .3-.3·i·2   1.9-1.9·i·2  π-π·i·2
   2-2·i·2     -.1--.1·i·2 .5-.5·i·2
:TRN(A-i·2·A)
[-2.5--2.5·-i·2  .3-.3·-i·2     2-2·-
  4.2-4.2·-i·2  1.9-1.9·-i·2 -.1--.
   2-2·-i·2       π-π·-i·2      .5-.5
```

If the argument is a real matrix, TRN simply produces the transpose of the real matrix. Try, for example, TRN(A), and compare it with TRAN(A).

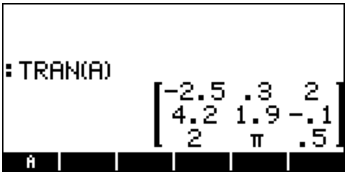
In RPN mode, the transconjugate of matrix **A** is calculated by using A TRN.

**Note**: The calculator also includes Function TRAN in the MATRICES/OPERATIONS sub-menu:





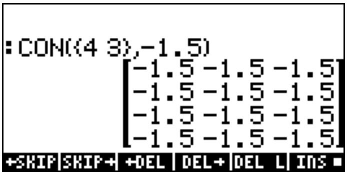
For example, in ALG mode:



```
:TRAN(A)
           [-2.5 .3   2 ]
           [ 4.2 1.9 -.1]
           [ 2   π   .5 ]
```

## Function CON

The function takes as argument a list of two elements, corresponding to the number of row and columns of the matrix to be generated, and a constant value. Function CON generates a matrix with constant elements. For example, in ALG mode, the following command creates a 4×3 matrix whose elements are all equal to –1.5:

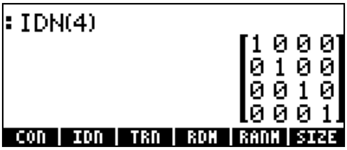


```
:CON({4 3},-1.5)
           [-1.5 -1.5 -1.5]
           [-1.5 -1.5 -1.5]
           [-1.5 -1.5 -1.5]
           [-1.5 -1.5 -1.5]
```

In RPN mode this is accomplished by using {4,3} ENTER 1 . 5 +/– ENTER CON.

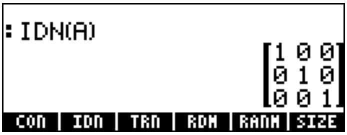
## Function IDN

Function IDN (IDeNtity matrix) creates an identity matrix given its size. Recall that an identity matrix has to be a square matrix, therefore, only one value is required to describe it completely. For example, to create a 4×4 identity matrix in ALG mode use:



```
:IDN(4)
                 [1 0 0 0]
                 [0 1 0 0]
                 [0 0 1 0]
                 [0 0 0 1]
```

You can also use an existing square matrix as the argument of function IDN, e.g.,



```
:IDN(A)
                   [1 0 0]
                   [0 1 0]
                   [0 0 1]
```

The resulting identity matrix will have the same dimensions as the argument matrix. Be aware that an attempt to use a rectangular (i.e., non-square) matrix as the argument of IDN will produce an error.

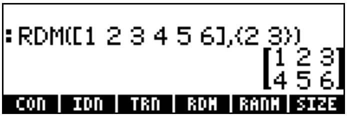
In RPN mode, the two exercises shown above are created by using: 4 ENTER IDN and ▓ IDN.

## Function RDM

Function RDM (Re-DiMensioning) is used to re-write vectors and matrices as matrices and vectors. The input to the function consists of the original vector or matrix followed by a list of a single number, if converting to a vector, or two numbers, if converting to a matrix. In the former case the number represents the vector's dimension, in the latter the number of rows and columns of the matrix. The following examples illustrate the use of function RDM:

### Re-dimensioning a vector into a matrix

The following example shows how to re-dimension a vector of 6 elements into a matrix of 2 rows and 3 columns in ALG mode:

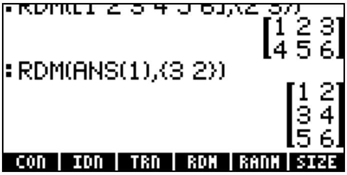


```
:RDM([1 2 3 4 5 6],{2 3})
                      [1 2 3]
                      [4 5 6]
```

In RPN mode, we can use `[1,2,3,4,5,6]` ENTER `{2,3}` ENTER `RDM` to produce the matrix shown above.

### Re-dimensioning a matrix into another matrix

In ALG mode, we now use the matrix created above and re-dimension it into a matrix of 3 rows and 2 columns:

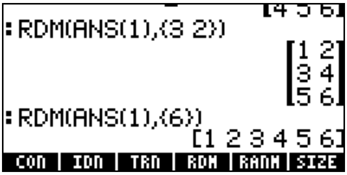


```
                      [1 2 3]
                      [4 5 6]
:RDM(ANS(1),{3 2})
                        [1 2]
                        [3 4]
                        [5 6]
```

In RPN mode, we simply use `{3,2}` ENTER `RDM`.

### Re-dimensioning a matrix into a vector

To re-dimension a matrix into a vector, we use as arguments the matrix followed by a list containing the number of elements in the matrix. For example, to convert the matrix from the previous example into a vector of length 6, in ALG mode, use:



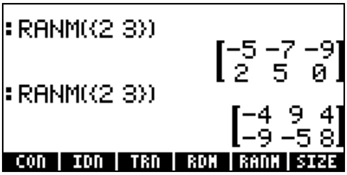
```
                          [4 5 6]
:RDM(ANS(1),{3 2})
                            [1 2]
                            [3 4]
                            [5 6]
:RDM(ANS(1),{6})
                    [1 2 3 4 5 6]
```

If using RPN mode, we assume that the matrix is in the stack and use {6} ENTER RDM.

**Note**: Function RDM provides a more direct and efficient way to transform lists to arrays and vice versa, than that provided at the end of chapter 9.

## Function RANM

Function RANM (RANdom Matrix) will generate a matrix with random integer elements given a list with the number of rows and columns (i.e., the dimensions of the matrix). For example, in ALG mode, two different 2×3 matrices with random elements are produced by using the same command, namely, RANM({2,3}) :



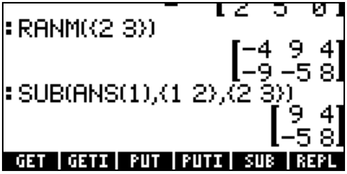
```
:RANM({2 3})
                     [-5 -7 -9]
                     [ 2  5  0]
:RANM({2 3})
                      [-4  9 4]
                      [-9 -5 8]
```

In RPN mode, use {2,3} ENTER RANM.

Obviously, the results you will get in your calculator will most certainly be different than those shown above. The random numbers generated are integer numbers uniformly distributed in the range [-10,10], i.e., each one of those 21 numbers has the same probability of being selected. Function RANM is useful for generating matrices of any size to illustrate matrix operations, or the application of matrix functions.

## Function SUB

Function SUB extracts a sub-matrix from an existing matrix, provided you indicate the initial and final position of the sub-matrix. For example, if we want to extract elements $a_{12}$, $a_{13}$, $a_{22}$, and $a_{23}$ from the last result, as a 2×2 sub-matrix, in ALG mode, use:

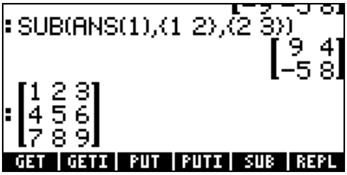


```
:RANM({2 3})
                 [-4  9 4]
                 [-9 -5 8]
:SUB(ANS(1),{1 2},{2 3})
                    [ 9 4]
                    [-5 8]
```

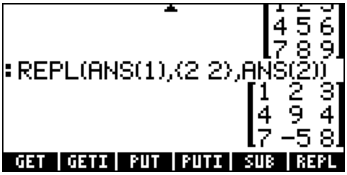
In RPN mode, assuming that the original 2×3 matrix is already in the stack, use {1,2} ENTER {2,3} ENTER SUB.

## Function REPL

Function REPL replaces or inserts a sub-matrix into a larger one. The input for this function is the matrix where the replacement will take place, the location where the replacement begins, and the matrix to be inserted. For example, keeping the matrix that we inherited from the previous example, enter the matrix: [[1,2,3],[4,5,6],[7,8,9]] . In ALG mode, the following screen shot to the left shows the new matrix before pressing ENTER. The screen shot to the right shows the application of function RPL to replace the matrix in ANS(2), the 2×2 matrix, into the 3×3 matrix currently located in ANS(1), starting at position {2,2}:



```
:SUB(ANS(1),{1 2},{2 3})
                    [ 9 4]
                    [-5 8]
 [1 2 3]
:[4 5 6]
 [7 8 9]
```



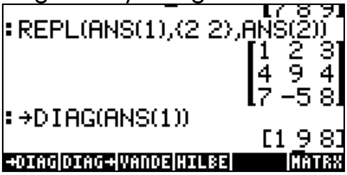
```
                   [4 5 6]
                   [7 8 9]
:REPL(ANS(1),{2 2},ANS(2))
                   [1  2 3]
                   [4  9 4]
                   [7 -5 8]
```

If working in the RPN mode, assuming that the 2×2 matrix was originally in the stack, we proceed as follows:

[[1,2,3],[4,5,6],[7,8,9]] ENTER ▶ (this last key swaps the contents of stack levels 1 and 2) {1,2} ENTER ▶ (another swapping of levels 1 and 2) REPL.

## Function →DIAG

Function →DIAG takes the main diagonal of a square matrix of dimensions n×n, and creates a vector of dimension n containing the elements of the main diagonal. For example, for the matrix remaining from the previous exercise, we can extract its main diagonal by using:



```
                                  [7 8 9]
:REPL(ANS(1),{2 2},ANS(2))
                                 [1  2 3]
                                 [4  9 4]
                                 [7 -5 8]
:→DIAG(ANS(1))
                                  [1 9 8]
```

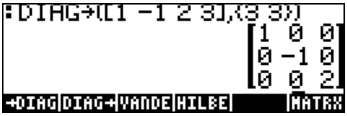
In RPN mode, with the 3×3 matrix in the stack, we simply have to activate function →DIAG to obtain the same result as above.

## Function DIAG→

Function DIAG→ takes a vector and a list of matrix dimensions {rows, columns}, and creates a diagonal matrix with the main diagonal replaced with the proper vector elements. For example, the command

DIAG→([1,-1,2,3],{3,3})

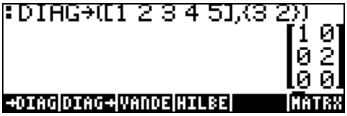
produces a diagonal matrix with the first 3 elements of the vector argument:



```
:DIAG→([1 -1 2 3],{3 3})
                                 [1  0 0]
                                 [0 -1 0]
                                 [0  0 2]
```

In RPN mode, we can use [1,-1,2,3] ENTER {3,3} ENTER DIAG→ to obtain the same result as above.

Another example of application of the DIAG→ function follows, in ALG mode:



```
:DIAG→([1 2 3 4 5],{3 2})
                                    [1 0]
                                    [0 2]
                                    [0 0]
```

In RPN mode, use [1,2,3,4,5] ENTER {3,2} ENTER DIAG→ .

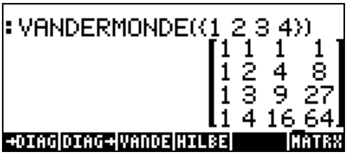
In this case a 3×2 matrix was to be created using as main diagonal elements as many elements as possible form the vector [1,2,3,4,5]. The main diagonal, for a rectangular matrix, starts at position (1,1) and moves on to position (2,2), (3,3), etc. until either the number of rows or columns is exhausted. In this case, the number of columns (2) was exhausted before the number of rows (3), so the main diagonal included only the elements in positions (1,1) and (2,2). Thus, only the first two elements of the vector were required to form the main diagonal.

## Function VANDERMONDE

Function VANDERMONDE generates the Vandermonde matrix of dimension n based on a given list of input data. The dimension n is, of course, the length of the list. If the input list consists of objects $\{x_1, x_2, \ldots x_n\}$, then, a Vandermonde matrix in the calculator is a matrix made of the following elements:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix}$$

For example, the following command in ALG mode for the list {1,2,3,4}:



```
:VANDERMONDE({1 2 3 4})
                    [1 1  1  1]
                    [1 2  4  8]
                    [1 3  9 27]
                    [1 4 16 64]
```

In RPN mode, enter `{1,2,3,4}` ENTER `VANDERMONDE`.

## Function HILBERT

Function HILBERT creates the Hilbert matrix corresponding to a dimension n. By definition, the n×n Hilbert matrix is $\mathbf{H}_n = [h_{jk}]_{n\times n}$, so that

$$h_{jk} = \frac{1}{j+k-1}$$

The Hilbert matrix has application in numerical curve fitting by the method of linear squares.

**A program to build a matrix out of a number of lists**

In this section we provide a couple of UserRPL programs to build a matrix out of a number of lists of objects. The lists may represent columns of the matrix (program CRMC) or rows of the matrix (program CRMR). The programs are entered with the calculator set to RPN mode, and the instructions for the keystrokes are given for system flag 117 set to SOFT menus. This section is intended for you to practice accessing programming functions in the calculator. The programs are listed below showing, in the left-hand side, the keystrokes necessary to enter the program steps, and, in the right-hand side, the characters entered in the display as you perform those keystrokes. First, we present the steps necessary to produce program CRMC.

## Lists represent columns of the matrix

The program CRMC allows you to put together a p×n matrix (i.e., p rows, n columns) out of n lists of p elements each. To create the program enter the following keystrokes:

| Keystroke sequence: | Produces: |
|---|---|
| ↱ « » | « |
| ↰ PRG STACK DUP | DUP |
| ↱ → SPC ALPHA ↰ N | → n |
| ↱ « » | << |
| 1 ↰ PRG STACK SWAP | 1 SWAP |
| ↰ PRG BRCH FOR FOR | FOR |
| ALPHA ↰ J | j |
| ↰ PRG TYPE OBJ→ | OBJ→ |
| →ARRY | →ARRY |
| ↰ PRG BRCH IF IF | IF |
| ALPHA ↰ J SPC | j |
| ALPHA ↰ N | n |
| ↰ PRG TEST < | < |
| ↰ PRG BRCH IF THEN | THEN |
| ALPHA ↰ J SPC 1 + | j 1 + |
| ↰ PRG STACK NXT ROLL | ROLL |
| ↰ PRG BRCH IF END | END |

| | |
|---|---|
| ← PRG BRCH FOR NEXT | NEXT |
| ← PRG BRCH IF IF | IF |
| ALPHA ← N SPC 1 | n 1 |
| ← PRG TEST > | > |
| ← PRG BRCH IF THEN | THEN |
| 1 SPC | 1 |
| ALPHA ← N SPC 1 − | n 1 - |
| ← PRG BRCH FOR FOR | FOR |
| ALPHA ← J SPC | j |
| ALPHA ← J SPC 1 + | j 1 + |
| ← PRG STACK NXT ROLL | ROLL |
| ← PRG BRCH FOR NEXT | NEXT |
| ← PRG BRCH IF END | END |
| ALPHA ← N SPC | n |
| ← MTH MATRX COL COL→ | COL→ |
| ENTER | Program is displayed in level 1 |

To save the program: ' ALPHA ALPHA C R M C ALPHA STO▶

**Note**: if you save this program in your HOME directory it will be available from any other sub-directory you use.

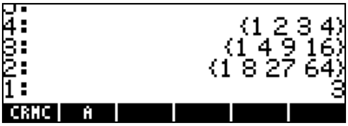
To see the contents of the program use VAR → CRMC. The program listing is the following:

```
« DUP → n « 1 SWAP FOR j OBJ→ →ARRY IF j n < THEN j 1 +
 ROLL END NEXT IF n 1 > THEN 1 n 1 - FOR j j 1 + ROLL
 NEXT END n COL→ » »
```

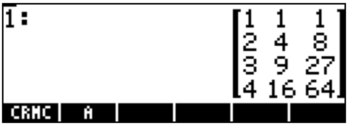
To use this program, in RPN mode, enter the n lists in the order that you want them as columns of the matrix, enter the value of n, and press CRMC. As an example, try the following exercise:

{1,2,3,4} ENTER {1,4,9,16} ENTER {1,8,27,64} ENTER 3 ENTER CRMC

The following screen shots show the RPN stack before and after running program CRMC:



```
4:            {1 2 3 4}
3:           {1 4 9 16}
2:          {1 8 27 64}
1:                    3
```

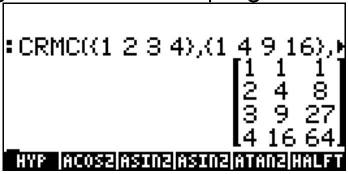


```
1:           [1  1  1]
             [2  4  8]
             [3  9 27]
             [4 16 64]
```

To use the program in ALG mode, press CRMC followed by a set of parentheses (↰ () ). Within the parentheses type the lists of data representing the columns of the matrix, separated by commas, and finally, a comma, and the number of columns. The command should look like this:

```
CRMC({1,2,3,4}, {1,4,9,16}, {1,8,27,64}, 3)
```

The ALG screen showing the execution of program CRMC is shown below:



```
:CRMC({1 2 3 4},{1 4 9 16},▸
                    ⎡1  1  1⎤
                    ⎢2  4  8⎥
                    ⎢3  9 27⎥
                    ⎣4 16 64⎦
```

## Lists represent rows of the matrix

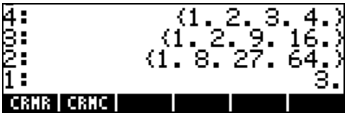
The previous program can be easily modified to create a matrix when the input lists will become the rows of the resulting matrix. The only change to be performed is to change COL→ for ROW→ in the program listing. To perform this change use:

| | |
|---|---|
| ↱ CRMC | List program CRMC in stack |
| ▼ ↱ ▼ ▲ ◀ ◀ ◀ | Move to end of program |
| ⬅ ⬅ ⬅ | Delete COL |
| ALPHA ALPHA R O W ALPHA ENTER | Type in ROW, enter program |

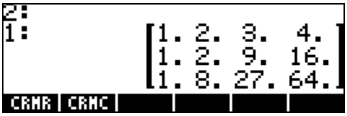
To store the program use: ' ALPHA ALPHA C R M R ALPHA STO▸

{1,2,3,4} ENTER {1,4,9,16} ENTER {1,8,27,64} ENTER 3 ENTER CRMR

The following screen shots show the RPN stack before and after running program CRMR:



```
4:           {1. 2. 3. 4.}
3:          {1. 2. 9. 16.}
2:        {1. 8. 27. 64.}
1:                      3.
```

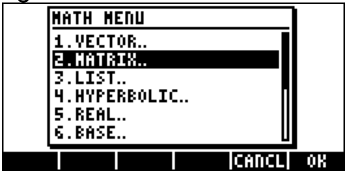


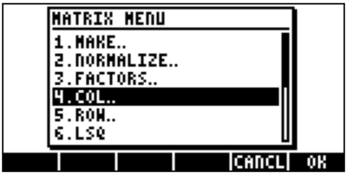
```
2:
1:        ⎡1. 2.  3.  4.⎤
          ⎢1. 2.  9. 16.⎥
          ⎣1. 8. 27. 64.⎦
```

These programs can be useful for statistical applications, specifically to create the statistical matrix ΣDAT. Examples of the use of these program are shown in a latter chapters.

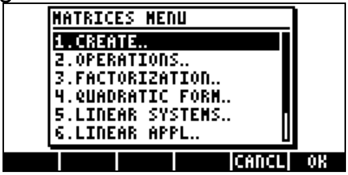
## Manipulating matrices by columns

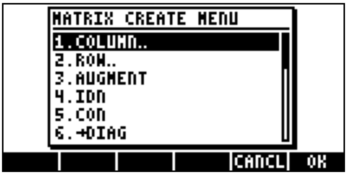
The calculator provides a menu with functions for manipulating matrices by operating in their columns. This menu is available through the MTH/MATRIX/COL.. sequence: (← MTH) shown in the figure below with system flag 117 set to CHOOSE boxes:



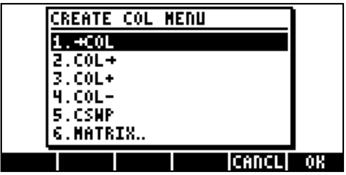


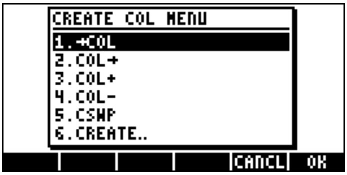
or through the MATRICES/CREATE/COLUMN sub-menu:



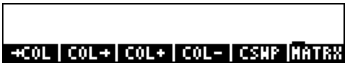


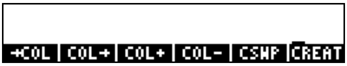
Both approaches will show the same functions:





When system flag 117 is set to SOFT menus, the COL menu is accessible through ← MTH MATRX MAKE COL , or through ← MATRICES CREAT COL . Both approaches will show the same set of functions:

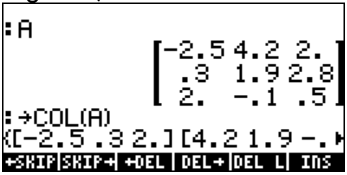




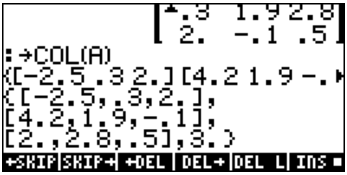
The operation of these functions is presented below.

### Function →COL

Function →COL takes as argument a matrix and decomposes it into vectors corresponding to its columns. An application of function →COL in ALG mode is shown below. The matrix used has been stored earlier in variable A. The matrix is shown in the figure to the left. The figure to the right shows the matrix decomposed in columns. To see the full result, use the line editor (triggered by pressing ▼).

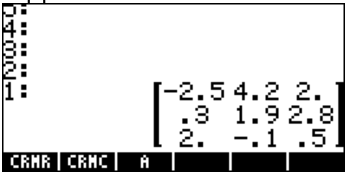


```
:A
              [-2.5 4.2 2. ]
              [ .3  1.9 2.8]
              [ 2.  -.1 .5 ]
:→COL(A)
{[-2.5 .3 2.] [4.2 1.9 -.▸
```

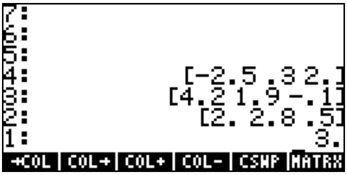


```
              [▲.3  1.9 2.8]
              [ 2.  -.1 .5 ]
:→COL(A)
{[-2.5 .3 2.] [4.2 1.9 -.▸
{[-2.5,.3,2.],
[4.2,1.9,-.1],
[2.,2.8,.5],3.}
```

In RPN mode, you need to list the matrix in the stack, and the activate function →COL, i.e., A →COL. The figure below shows the RPN stack before and after the application of function →COL.



```
5:
4:
3:
2:
1:            [-2.5 4.2 2. ]
              [ .3  1.9 2.8]
              [ 2.  -.1 .5 ]
```



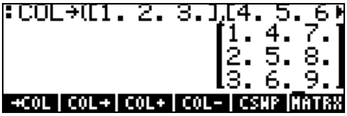
```
7:
6:
5:
4:           [-2.5 .3 2.]
3:          [4.2 1.9 -.1]
2:             [2. 2.8 .5]
1:                      3.
```

In this result, the first column occupies the highest stack level after decomposition, and stack level 1 is occupied by the number of columns of the original matrix. The matrix does not survive decomposition, i.e., it is no longer available in the stack.

## Function COL→

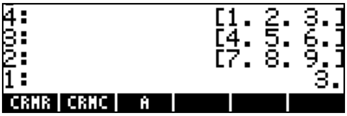
Function COL→ has the opposite effect of Function →COL, i.e., given n vectors of the same length, and the number n, function COL→ builds a matrix by placing the input vectors as columns of the resulting matrix. Here is an example in ALG mode. The command used was:

```
COL→([1,2,3],[4,5,6],[7,8,9],3)
```

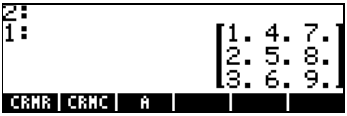


```
:COL→([1. 2. 3.],[4. 5. 6▸
                 [1. 4. 7.]
                 [2. 5. 8.]
                 [3. 6. 9.]
```

In RPN mode, place the n vectors in stack levels n+1, n, n-1,…,2, and the number n in stack level 1. With this set up, function COL→ places the vectors as columns in the resulting matrix. The following figure shows the RPN stack before and after using function COL→.



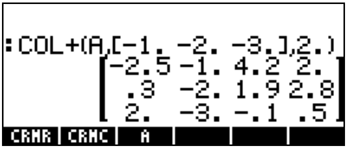
```
4:             [1. 2. 3.]
3:             [4. 5. 6.]
2:             [7. 8. 9.]
1:                     3.
```



```
2:
1:             [1. 4. 7.]
               [2. 5. 8.]
               [3. 6. 9.]
```

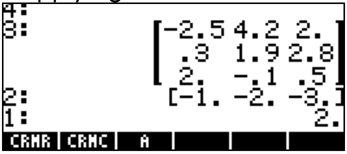
## Function COL+

Function COL+ takes as argument a matrix, a vector with the same length as the number of rows in the matrix, and an integer number n representing the location of a column. Function COL+ inserts the vector in column n of the matrix. For example, in ALG mode, we'll insert the second column in matrix A with the vector [-1,-2,-3], i.e.,

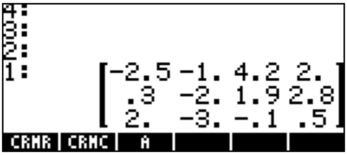


```
:COL+(A,[-1. -2. -3.],2.)
       [-2.5 -1. 4.2 2. ]
       [ .3  -2. 1.9 2.8]
       [ 2.  -3. -.1 .5 ]
```

In RPN mode, enter the matrix first, then the vector, and the column number, before applying function COL+. The figure below shows the RPN stack before and after applying function COL+.



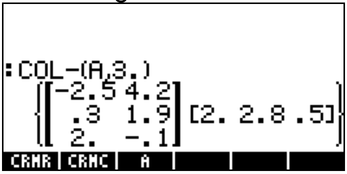
```
4:
3:          [-2.5 4.2 2. ]
            [ .3  1.9 2.8]
            [ 2.  -.1 .5 ]
2:           [-1. -2. -3.]
1:                      2.
```



```
4:
3:
2:
1:     [-2.5 -1. 4.2 2. ]
       [ .3  -2. 1.9 2.8]
       [ 2.  -3. -.1 .5 ]
```

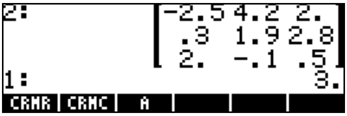
## Function COL-

Function COL- takes as argument a matrix and an integer number representing the position of a column in the matrix. Function returns the original matrix minus a column, as well as the extracted column shown as a vector. Here is an example in the ALG mode using the matrix stored in A:

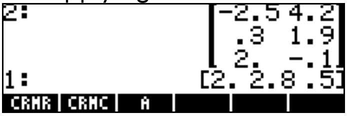


```
:COL-(A,3.)
  {[-2.5 4.2]              }
   [ .3  1.9] [2. 2.8 .5]
   [ 2.  -.1]
```

In RPN mode, place the matrix in the stack first, then enter the number representing a column location before applying function COL-.  The following figure shows the RPN stack before and after applying function COL-.



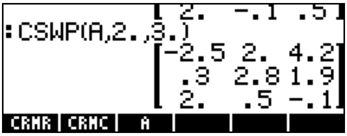
```
2:          [-2.5 4.2 2. ]
            [ .3  1.9 2.8]
            [ 2.  -.1 .5 ]
1:                      3.
```



```
2:            [-2.5 4.2]
              [ .3  1.9]
              [ 2.  -.1]
1:          [2. 2.8 .5]
```

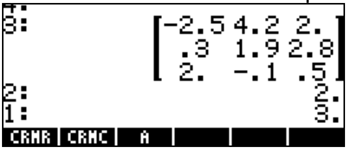
## Function CSWP

Function CSWP (Column SWaP) takes as arguments two indices, say, i and j, (representing two distinct columns in a matrix), and a matrix, and produces a new matrix with columns i and j swapped.   The following example, in ALG mode, shows an application of this function.  We use the matrix stored in variable A for the example.  This matrix is listed first.

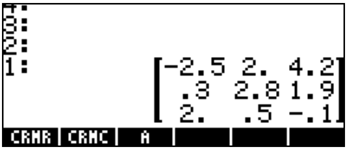


```
            [ 2.   -.1  .5 ]
:CSWP(A,2.,3.)
            [-2.5 2.  4.2]
            [ .3  2.8 1.9]
            [ 2.   .5 -.1]
```

In RPN mode, function CSWP lets you swap the columns of a matrix listed in stack level 3, whose indices are listed in stack levels 1 and 2. For example, the following figure shows the RPN stack before and after applying function CSWP to matrix A in order to swap columns 2 and 3:



```
3:          [-2.5 4.2 2. ]
            [ .3  1.9 2.8]
            [ 2.  -.1 .5 ]
2:                      2.
1:                      3.
```



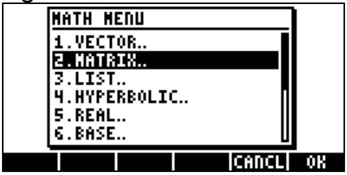
```
3:
2:
1:          [-2.5 2.  4.2]
            [ .3  2.8 1.9]
            [ 2.   .5 -.1]
```

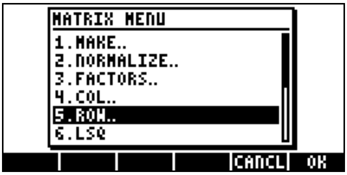
As you can see, the columns that originally occupied positions 2 and 3 have been swapped.  Swapping of columns, and of rows (see below), is commonly used when solving systems of linear equations with matrices.  Details of these operations will be given in a subsequent Chapter.

# Manipulating matrices by rows

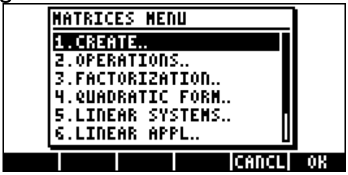
The calculator provides a menu with functions for manipulating matrices by operating in their rows.  This menu is available through the

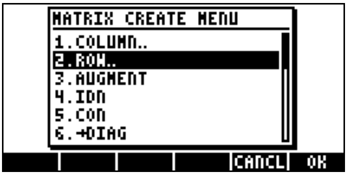
MTH/MATRIX/ROW.. sequence: (← MTH) shown in the figure below with system flag 117 set to CHOOSE boxes:



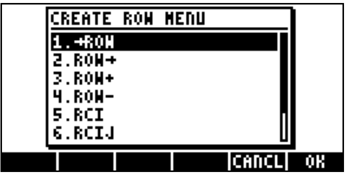


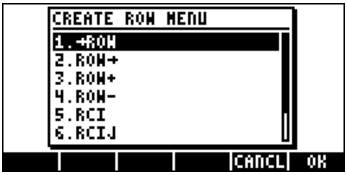
or through the MATRICES/CREATE/ROW sub-menu:



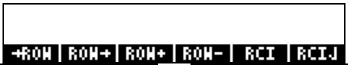


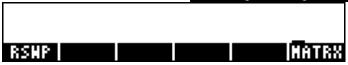
Both approaches will show the same functions:

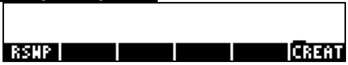




When system flag 117 is set to SOFT menus, the ROW menu is accessible through ← MTH MATRX MAKE ROW , or through ← MATRICES CREAT ROW . Both approaches will show the same set of functions:



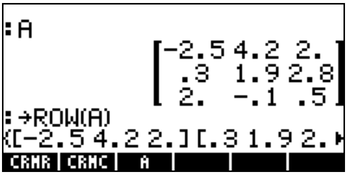




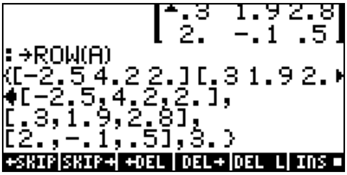
The operation of these functions is presented below.

## Function →ROW

Function →ROW takes as argument a matrix and decomposes it into vectors corresponding to its rows. An application of function →ROW in ALG mode is shown below. The matrix used has been stored earlier in variable A. The matrix is shown in the figure to the left. The figure to the right shows the matrix decomposed in rows. To see the full result, use the line editor (triggered by pressing ▽ ).

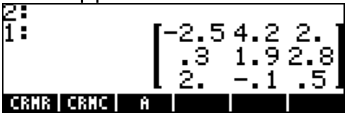


```
:A
          [-2.5 4.2 2. ]
          [ .3  1.9 2.8]
          [ 2.  -.1 .5 ]
:→ROW(A)
{[-2.5 4.2 2.][.3 1.9 2.▸
```

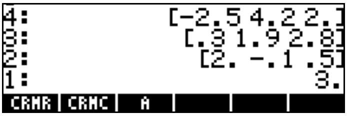


```
          [ .3  1.9 2.8]
          [ 2.  -.1 .5 ]
:→ROW(A)
{[-2.5 4.2 2.][.3 1.9 2.▸
◂[-2.5,4.2,2.],
[.3,1.9,2.8],
[2.,-.1,.5],3.}
```

In RPN mode, you need to list the matrix in the stack, and the activate function →ROW, i.e., →ROW. The figure below shows the RPN stack before and after the application of function →ROW.



```
2:
1:        [-2.5 4.2 2. ]
          [ .3  1.9 2.8]
          [ 2.  -.1 .5 ]
```



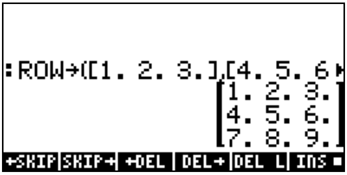
```
4:          [-2.5 4.2 2.]
3:            [.3 1.9 2.8]
2:             [2. -.1 .5]
1:                      3.
```

In this result, the first row occupies the highest stack level after decomposition, and stack level 1 is occupied by the number of rows of the original matrix. The matrix does not survive decomposition, i.e., it is no longer available in the stack.

## Function ROW→

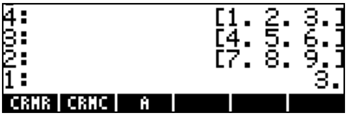
Function ROW→ has the opposite effect of the function →ROW, i.e., given n vectors of the same length, and the number n, function ROW→ builds a matrix by placing the input vectors as rows of the resulting matrix. Here is an example in ALG mode. The command used was:

```
ROW→([1,2,3],[4,5,6],[7,8,9],3)
```

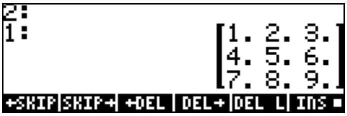


```
:ROW→([1. 2. 3.],[4. 5. 6▸
                [1. 2. 3.]
                [4. 5. 6.]
                [7. 8. 9.]
```

In RPN mode, place the n vectors in stack levels n+1, n, n-1,…,2, and the number n in stack level 1. With this set up, function ROW→ places the vectors as rows in the resulting matrix. The following figure shows the RPN stack before and after using function ROW→.



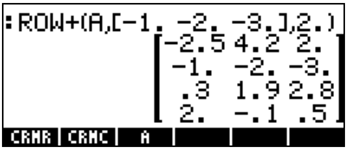
```
4:                [1. 2. 3.]
3:                [4. 5. 6.]
2:                [7. 8. 9.]
1:                        3.
```



```
2:
1:
                  [1. 2. 3.]
                  [4. 5. 6.]
                  [7. 8. 9.]
```

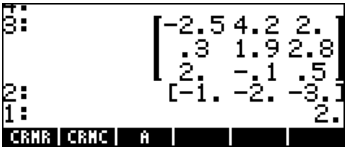
## Function ROW+

Function ROW+ takes as argument a matrix, a vector with the same length as the number of rows in the matrix, and an integer number n representing the location of a row. Function ROW+ inserts the vector in row n of the matrix. For example, in ALG mode, we'll insert the second row in matrix A with the vector [-1,-2,-3], i.e.,

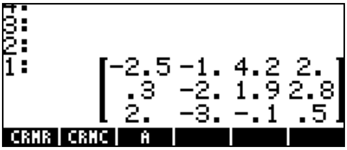


```
:ROW+(A,[-1. -2. -3.],2.)
            [-2.5 4.2 2. ]
            [ -1. -2. -3.]
            [ .3  1.9 2.8]
            [ 2.  -.1 .5 ]
```

In RPN mode, enter the matrix first, then the vector, and the row number, before applying function ROW+. The figure below shows the RPN stack before and after applying function ROW+.



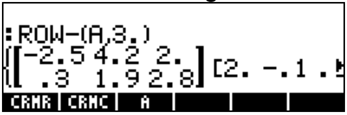
```
4:
3:           [-2.5 4.2 2. ]
             [ .3  1.9 2.8]
             [ 2.  -.1 .5 ]
2:            [-1. -2. -3.]
1:                       2.
```



```
4:
3:
2:
1:     [-2.5 -1. 4.2 2. ]
       [ .3  -2. 1.9 2.8]
       [ 2.  -3. -.1 .5 ]
```

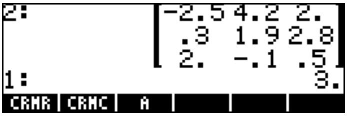
## Function ROW-

Function ROW- takes as argument a matrix and an integer number representing the position of a row in the matrix. The function returns the original matrix, minus a row, as well as the extracted row shown as a vector. Here is an example in the ALG mode using the matrix stored in A:

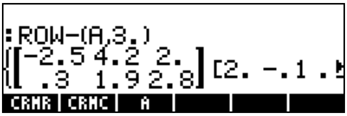


```
:ROW-(A,3.)
{[-2.5 4.2 2. ]
 [ .3  1.9 2.8] [2. -.1 .▸
```

In RPN mode, place the matrix in the stack first, then enter the number representing a row location before applying function ROW-. The following figure shows the RPN stack before and after applying function ROW-.



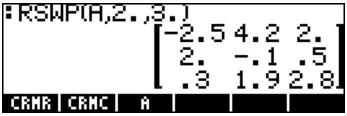
```
2:          ┌-2.5 4.2 2. ┐
            │ .3  1.9 2.8│
            └ 2.  -.1  .5┘
1:                     3.
```



```
:ROW-(A,3.)
{┌-2.5 4.2 2. ┐ [2. -.1 .▸
 └ .3  1.9 2.8┘
```

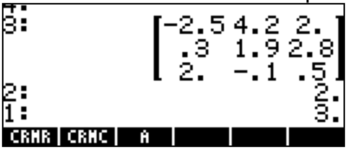
## Function RSWP

Function RSWP (Row SWaP) takes as arguments two indices, say, i and j, (representing two distinct rows in a matrix), and a matrix, and produces a new matrix with rows i and j swapped. The following example, in ALG mode, shows an application of this function. We use the matrix stored in variable A for the example. This matrix is listed first.

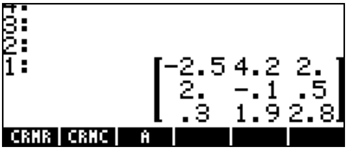


```
:RSWP(A,2.,3.)
            ┌-2.5 4.2 2. ┐
            │ 2.  -.1  .5│
            └ .3  1.9 2.8┘
```

In RPN mode, function CSWP lets you swap the rows of a matrix listed in stack level 3, whose indices are listed in stack levels 1 and 2. For example, the following figure shows the RPN stack before and after applying function CSWP to matrix A in order to swap rows 2 and 3:



```
4:
3:          ┌-2.5 4.2 2. ┐
            │ .3  1.9 2.8│
            └ 2.  -.1  .5┘
2:                     2.
1:                     3.
```

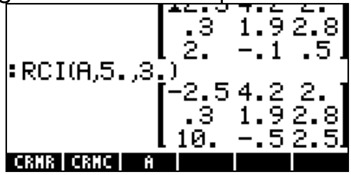


```
4:
3:
2:
1:
            ┌-2.5 4.2 2. ┐
            │ 2.  -.1  .5│
            └ .3  1.9 2.8┘
```

As you can see, the columns that originally occupied positions 2 and 3 have been swapped.

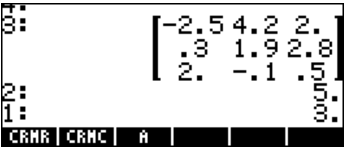
## Function RCI

Function RCI stands for multiplying *R*ow I by a *C*onstant value and replace the resulting row at the same location. The following example, written in ALG mode, takes the matrix stored in A, and multiplies the constant value 5 into row number 3, replacing the row with this product.

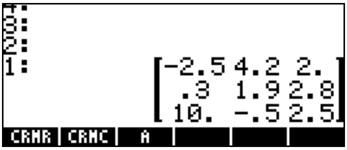


```
            │ .3  1.9 2.8│
            └ 2.  -.1  .5┘
:RCI(A,5.,3.)
            ┌-2.5 4.2 2. ┐
            │ .3  1.9 2.8│
            └ 10. -.5 2.5┘
```

This same exercise done in RPN mode is shown in the next figure. The left-hand side figure shows the setting up of the matrix, the factor and the row number, in stack levels 3, 2, and 1. The right-hand side figure shows the resulting matrix after function RCI is activated.



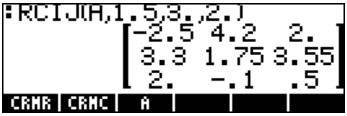
```
4:
3:          [-2.5 4.2 2. ]
            [ .3  1.9 2.8]
            [ 2.  -.1 .5 ]
2:                     5.
1:                     3.
```



```
4:
3:
2:
1:          [-2.5 4.2 2. ]
            [ .3  1.9 2.8]
            [ 10. -.5 2.5]
```

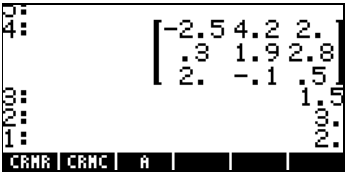
## Function RCIJ

Function RCIJ stands for "take Row I and multiplying it by a constant C and then add that multiplied row to row J, replacing row J with the resulting sum." This type of row operation is very common in the process of Gaussian or Gauss-Jordan elimination (more details on this procedure are presented in a subsequent Chapter). The arguments of the function are: (1) the matrix, (2) the constant value, (3) the row to be multiplied by the constant in(2), and (4) the row to be replaced by the resulting sum as described above. For example, taking the matrix stored in variable A, we are going to multiply column 3 times 1.5, and add it to column 2. The following example is performed in ALG mode:

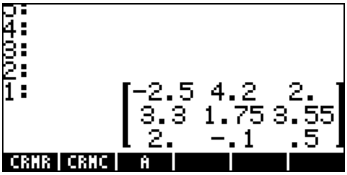


```
:RCIJ(A,1.5,3.,2.)
            [-2.5 4.2  2.  ]
            [ 3.3 1.75 3.55]
            [ 2.  -.1  .5  ]
```

In RPN mode, enter the matrix first, followed by the constant value, then by the row to be multiplied by the constant value, and finally enter the row that will be replaced. The following figure shows the RPN stack before and after applying function RCIJ under the same conditions as in the ALG example shown above:



```
5:
4:          [-2.5 4.2 2. ]
            [ .3  1.9 2.8]
            [ 2.  -.1 .5 ]
3:                    1.5
2:                     3.
1:                     2.
```



```
5:
4:
3:
2:
1:     [-2.5 4.2  2.  ]
       [ 3.3 1.75 3.55]
       [ 2.  -.1  .5  ]
```

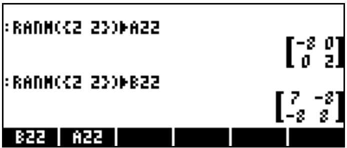
# Chapter 11
# Matrix Operations and Linear Algebra

In Chapter 10 we introduced the concept of a matrix and presented a number of functions for entering, creating, or manipulating matrices. In this Chapter we present examples of matrix operations and applications to problems of linear algebra.

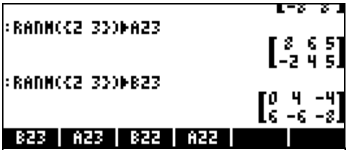
## Operations with matrices

Matrices, like other mathematical objects, can be added and subtracted. They can be multiplied by a scalar, or among themselves. An important operation for linear algebra applications is the inverse of a matrix. Details of these operations are presented next.

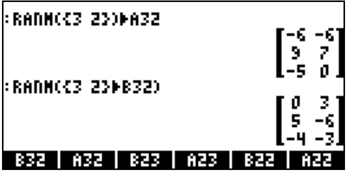
To illustrate the operations we will create a number of matrices that we will store in variables. The generic name of the matrices will be A*ij* and B*ij*, where *i* represents the number of rows and *j* the number of columns of the matrices. The matrices to be used are generated by using function RANM (random matrices). If you try this exercise in your calculator you will get different matrices than the ones listed herein, unless you store them into your calculator exactly as shown below. Here are the matrices A22, B22, A23, B23, A32, B32, A33 and B33 created in ALG mode:



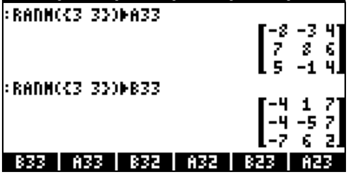
```
:RANM({2 2})▶A22
                         [-8 0]
                         [ 0 2]
:RANM({2 2})▶B22
                         [ 7 -8]
                         [-8  8]
```



```
:RANM({2 3})▶A23
                         [ 8 6 5]
                         [-2 4 5]
:RANM({2 3})▶B23
                         [0  4 -4]
                         [6 -6 -8]
```



```
:RANM({3 2})▶A32
                         [-6 -6]
                         [ 9  7]
                         [-5  0]
:RANM({3 2}▶B32)
                         [ 0  3]
                         [ 5 -6]
                         [-4 -3]
```



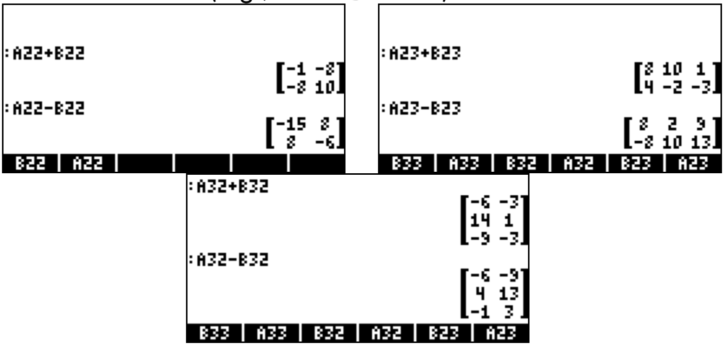
```
:RANM({3 3})▶A33
                         [-8 -3 4]
                         [ 7  8 6]
                         [ 5 -1 4]
:RANM({3 3})▶B33
                         [-4  1 7]
                         [-4 -5 7]
                         [-7  6 2]
```

In RPN mode, the steps to follow are:

```
{2,2} ENTER RANM 'A22' STO▶     {2,2} ENTER RANM 'B22' STO▶
{2,3} ENTER RANM 'A23' STO▶     {2,3} ENTER RANM 'B23' STO▶
{3,2} ENTER RANM 'A32' STO▶     {3,2} ENTER RANM 'B32' STO▶
{3,3} ENTER RANM 'A33' STO▶     {3,3} ENTER RANM 'B33' STO▶
```

## Addition and subtraction

Consider a pair of matrices **A** = $[a_{ij}]_{m\times n}$ and **B** = $[b_{ij}]_{m\times n}$. Addition and subtraction of these two matrices is only possible if they have the same number of rows and columns. The resulting matrix, **C** = **A** ± **B** = $[c_{ij}]_{m\times n}$ has elements $c_{ij} = a_{ij} \pm b_{ij}$. Some examples in ALG mode are shown below using the matrices stored above (e.g., A22 + B22)



```
:A22+B22
                [-1  -8]
                [-8  10]
:A22-B22
                [-15  8]
                [  8 -6]
 B22 | A22 |     |     |     |
```

```
:A23+B23
                [8 10  1]
                [4 -2 -3]
:A23-B23
                [ 8  2  9]
                [-8 10 13]
 B33 | A33 | B32 | A32 | B23 | A23
```

```
:A32+B32
                [-6 -3]
                [14  1]
                [-9 -3]
:A32-B32
                [-6 -9]
                [ 4 13]
                [-1  3]
 B33 | A33 | B32 | A32 | B23 | A23
```

In RPN mode, the steps to follow are:

```
A22 ENTER B22 ENTER +     A22 ENTER B22 ENTER −
A23 ENTER B23 ENTER +     A23 ENTER B23 ENTER −
A32 ENTER B32 ENTER +     A32 ENTER B32 ENTER −
```

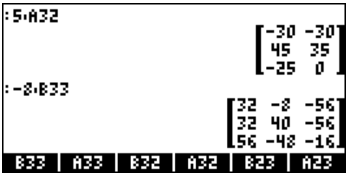
Translating the ALG examples to RPN is straightforward, as illustrated here. The remaining examples of matrix operations will be performed in ALG mode only.

## Multiplication

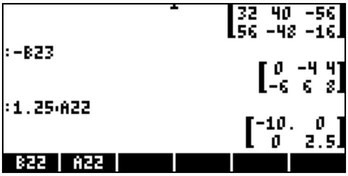
There are different multiplication operations that involve matrices. These are described next.

### Multiplication by a scalar

Multiplication of the matrix **A** = $[a_{ij}]_{m\times n}$ by a scalar k results in the matrix **C** = k**A** = $[c_{ij}]_{m\times n} = [ka_{ij}]_{m\times n}$. In particular, the negative of a matrix is defined by the operation -**A** =(-1)**A** = $[-a_{ij}]_{m\times n}$. Some examples of multiplication of a matrix by a scalar are shown below.

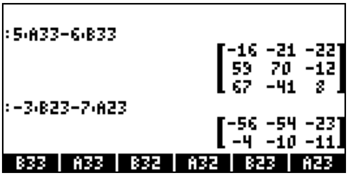


```
:5·A32
                    [-30 -30]
                    [ 45  35]
                    [-25   0]
:-8·B33
                 [32  -8 -56]
                 [32  40 -56]
                 [56 -48 -16]
```

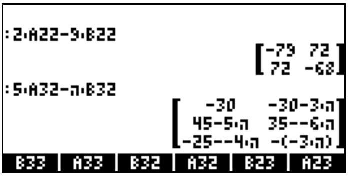


```
                 [32  40 -56]
                 [56 -48 -16]
:-B23
                    [ 0 -4 4]
                    [-6  6 8]
:1.25·A22
                  [-10.   0 ]
                  [  0   2.5]
```

By combining addition and subtraction with multiplication by a scalar we can form linear combinations of matrices of the same dimensions, e.g.,

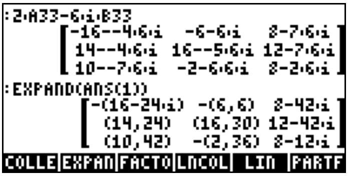


```
:5·A33-6·B33
                 [-16 -21 -22]
                 [ 59  70 -12]
                 [ 67 -41   8]
:-3·B23-7·A23
                 [-56 -54 -23]
                 [ -4 -10 -11]
```



```
:2·A22-9·B22
                    [-79  72]
                    [ 72 -68]
:5·A32-π·B32
        [   -30     -30-3·π ]
        [ 45-5·π    35--6·π ]
        [-25--4·π  -(-3·π)  ]
```

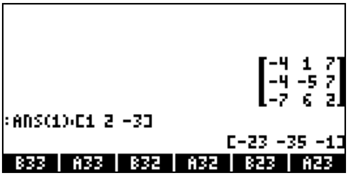
In a linear combination of matrices, we can multiply a matrix by an imaginary number to obtain a matrix of complex numbers, e.g.,



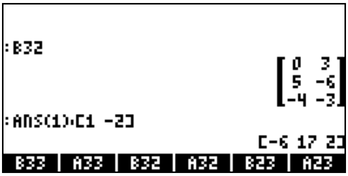
```
:2·A33-6·i·B33
     [-16--4·6·i   -6-6·i    8-7·6·i]
     [ 14--4·6·i 16--5·6·i 12-7·6·i]
     [ 10--7·6·i  -2-6·6·i   8-2·6·i]
:EXPAND(ANS(1))
       [-(16-24·i)  -(6,6)    8-42·i]
       [  (14,24)   (16,30) 12-42·i]
       [  (10,42)  -(2,36)   8-12·i]
```

### Matrix-vector multiplication

Matrix-vector multiplication is possible only if the number of columns of the matrix is equal to the length of the vector. This operation follows the rules of matrix multiplication as shown in the next section. A couple of examples of matrix-vector multiplication follow:



```
                      [-4  1 7]
                      [-4 -5 7]
                      [-7  6 2]
:ANS(1)·[1 2 -3]
                 [-23 -35 -1]
```



```
:B32
                        [ 0  3]
                        [ 5 -6]
                        [-4 -3]
:ANS(1)·[1 -2]
                     [-6 17 2]
```

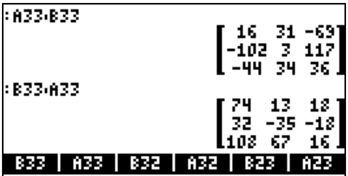
Vector-matrix multiplication, on the other hand, is not defined. This multiplication can be performed, however, as a special case of matrix multiplication as defined next.

### Matrix multiplication

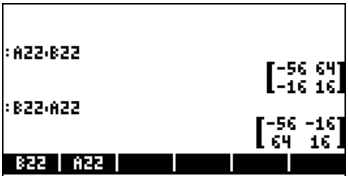
Matrix multiplication is defined by $\mathbf{C}_{m\times n} = \mathbf{A}_{m\times p}\cdot\mathbf{B}_{p\times n}$, where $\mathbf{A} = [a_{ij}]_{m\times p}$, $\mathbf{B} = [b_{ij}]_{p\times n}$, and $\mathbf{C} = [c_{ij}]_{m\times n}$. Notice that matrix multiplication is only possible if the number of columns in the first operand is equal to the number of rows of the second operand. The general term in the product, $c_{ij}$, is defined as

$$c_{ij} = \sum_{k=1}^{p} a_{ik}\cdot b_{kj},\ for\ i = 1,2,\ldots,m;\ j = 1,2,\ldots,n.$$

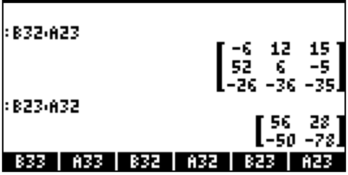
This is the same as saying that the element in the i-th row and j-th column of the product, **C**, results from multiplying term-by-term the i-th row of **A** with the j-th column of **B**, and adding the products together. Matrix multiplication is not commutative, i.e., in general, **A**·**B** ≠ **B**·**A**. Furthermore, one of the multiplications may not even exist. The following screen shots show the results of multiplications of the matrices that we stored earlier:



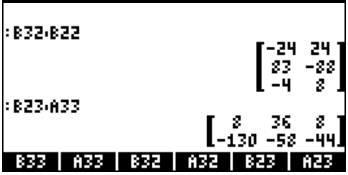
```
:A33·B33
                    [  16  31 -69]
                    [-102   3 117]
                    [ -44  34  36]
:B33·A33
                    [ 74  13  18]
                    [ 32 -35 -18]
                    [108  67  16]
```



```
:A22·B22
                    [-56 64]
                    [-16 16]
:B22·A22
                    [-56 -16]
                    [ 64  16]
```



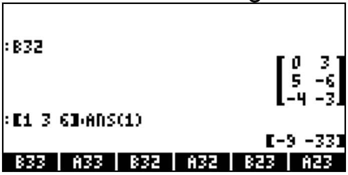
```
:B32·A23
                    [ -6  12  15]
                    [ 52   6  -5]
                    [-26 -36 -35]
:B23·A32
                    [ 56  28]
                    [-50 -78]
```



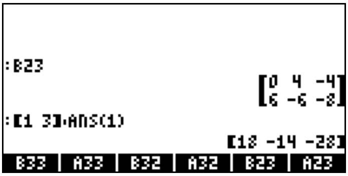
```
:B32·B22
                    [-24  24]
                    [ 83 -88]
                    [ -4   8]
:B23·A33
                    [   8  36   8]
                    [-130 -58 -44]
```

The matrix-vector multiplication introduced in the previous section can be thought of as the product of a matrix m×n with a matrix n×1 (i.e., a column vector) resulting in an m×1 matrix (i.e., another vector). To verify this assertion check the examples presented in the previous section. Thus, the vectors defined in Chapter 9 are basically column vectors for the purpose of matrix multiplication.

The product of a vector with a matrix is possible if the vector is a row vector, i.e., a 1×m matrix, which multiplied with a matrix m×n produces a 1xn matrix (another row vector). For the calculator to identify a row vector, you must use double brackets to enter it, e.g.,



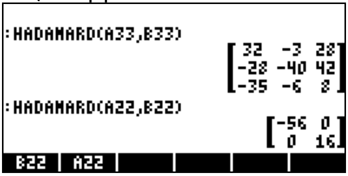
```
:B32
                         [ 0  3]
                         [ 5 -6]
                         [-4 -3]
:[[1 3 6]]·ANS(1)
                        [-9 -33]
 B33 | A33 | B32 | A32 | B23 | A23
```



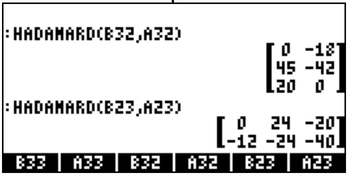
```
:B23
                      [0  4 -4]
                      [6 -6 -8]
:[[1 3]]·ANS(1)
                    [18 -14 -28]
 B33 | A33 | B32 | A32 | B23 | A23
```

### Term-by-term multiplication

Term-by-term multiplication of two matrices of the same dimensions is possible through the use of function HADAMARD. The result is, of course, another matrix of the same dimensions. This function is available through Function catalog (↱ CAT ), or through the MATRICES/OPERATIONS sub-menu (↰ MATRICES ). Applications of function HADAMARD are presented next:



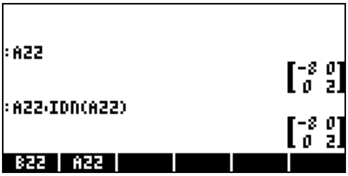
```
:HADAMARD(A33,B33)
                    [ 32  -3 28]
                    [-28 -40 42]
                    [-35  -6  8]
:HADAMARD(A22,B22)
                       [-56  0]
                       [  0 16]
 B22 | A22 |     |     |     |
```



```
:HADAMARD(B32,A32)
                       [ 0 -18]
                       [45 -42]
                       [20   0]
:HADAMARD(B23,A23)
                  [  0  24 -20]
                  [-12 -24 -40]
 B33 | A33 | B32 | A32 | B23 | A23
```

### The identity matrix

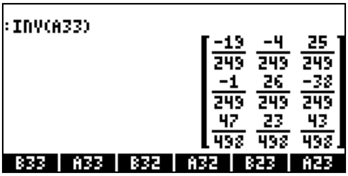
In Chapter 9 we introduce the identity matrix as the matrix **I** = $[\delta_{ij}]_{n\times n}$, where $\delta_{ij}$ is the Kronecker's delta function. Identity matrices can be obtained by using function IDN described in Chapter 9. The identity matrix has the property that **A·I** = **I·A** = **A**. To verify this property we present the following examples using the matrices stored earlier on:



```
:A22
                         [-8 0]
                         [ 0 2]
:A22·IDN(A22)
                         [-8 0]
                         [ 0 2]
 B22 | A22 |     |     |     |
```

### The inverse matrix

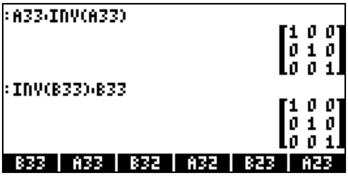
The inverse of a square matrix **A** is the matrix $\mathbf{A}^{-1}$ such that $\mathbf{A}\cdot\mathbf{A}^{-1} = \mathbf{A}^{-1}\cdot\mathbf{A} = \mathbf{I}$, where **I** is the identity matrix of the same dimensions as **A**. The inverse of a matrix is obtained in the calculator by using the inverse function, INV (i.e., the 1/x key). An example of the inverse of one of the matrices stored earlier is presented next:



:INV(A33)

$$\begin{bmatrix} \frac{-19}{249} & \frac{-4}{249} & \frac{25}{249} \\ \frac{-1}{249} & \frac{26}{249} & \frac{-38}{249} \\ \frac{47}{498} & \frac{23}{498} & \frac{43}{498} \end{bmatrix}$$

To verify the properties of the inverse matrix, we present the following multiplications:

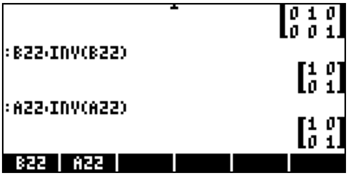


:A33·INV(A33)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

:INV(B33)·B33

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}\Bigg]$$

:B22·INV(B22)

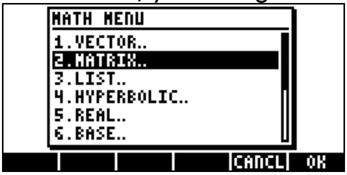
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

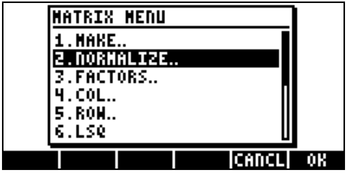
:A22·INV(A22)

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

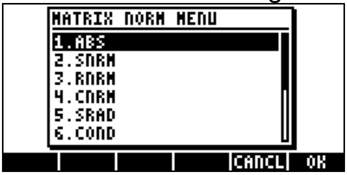
## Characterizing a matrix (The matrix NORM menu)

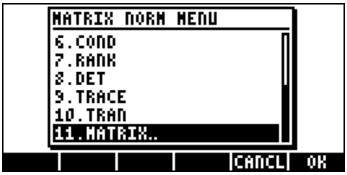
The matrix NORM (NORMALIZE) menu is accessed through the keystroke sequence ← MTH (system flag 117 set to CHOOSE boxes):





This menu contains the following functions:





These functions are described next. Because many of these functions use concepts of matrix theory, such as singular values, rank, etc., we will include short descriptions of these concepts intermingled with the description of functions.

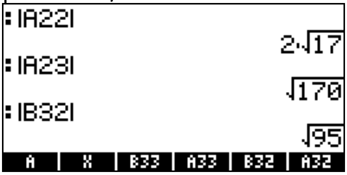
## Function ABS

Function ABS calculates what is known as the Frobenius norm of a matrix. For a matrix **A** = $[a_{ij}]_{m\times n}$, the Frobenius norm of the matrix is defined as

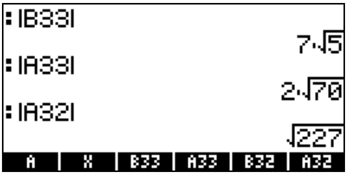
$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}^2}$$

If the matrix under consideration in a row vector or a column vector, then the Frobenius norm , $||\mathbf{A}||_F$ , is simply the vector's magnitude. Function ABS is accessible directly in the keyboard as ↰ ABS .

Try the following exercises in ALG mode (using the matrices stored earlier for matrix operations):



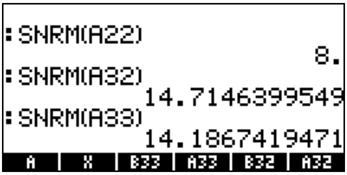
```
:|A22|
                2·√17
:|A23|
                 √170
:|B32|
                  √95
```



```
:|B33|
                 7·√5
:|A33|
                2·√70
:|A32|
                 √227
```

## Function SNRM

Function SNRM calculates the Spectral NoRM of a matrix, which is defined as the matrix's largest singular value, also known as the Euclidean norm of the matrix. For example,



```
:SNRM(A22)
                   8.
:SNRM(A32)
        14.7146399549
:SNRM(A33)
        14.1867419471
```

**Singular value decomposition**

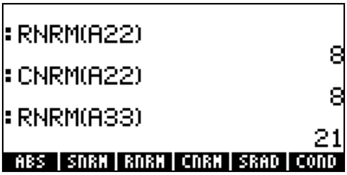
To understand the operation of Function SNRM, we need to introduce the concept of matrix decomposition. Basically, matrix decomposition involves the determination of two or more matrices that, when multiplied in a certain order (and, perhaps, with some matrix inversion or transposition thrown in), produce the original matrix. The Singular Value Decomposition (SVD) is such that a rectangular matrix $\mathbf{A}_{m\times n}$ is written as $\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m} \cdot \mathbf{S}_{m\times n} \cdot \mathbf{V}^{T}_{n\times n}$,

Where **U** and V are orthogonal matrices, and **S** is a diagonal matrix. The diagonal elements of **S** are called the singular values of A and are usually ordered so that $s_i \geq s_{i+1}$, for i = 1, 2, …, n-1. The columns [$\mathbf{u}_j$] of **U** and [$\mathbf{v}_j$] of **V** are the corresponding singular vectors. (Orthogonal matrices are such that $\mathbf{U}\cdot \mathbf{U}^T = \mathbf{I}$. A diagonal matrix has non-zero elements only along its main diagonal).

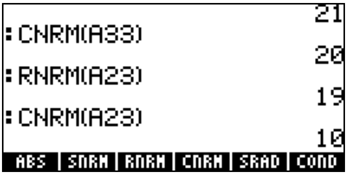
The rank of a matrix can be determined from its SVD by counting the number of non-singular values. Examples of SVD will be presented in a subsequent section.

## Functions RNRM and CNRM

Function RNRM returns the Row NoRM of a matrix, while function CNRM returns the Column NoRM of a matrix. Examples,



```
:RNRM(A22)
                              8
:CNRM(A22)
                              8
:RNRM(A33)
                             21
```



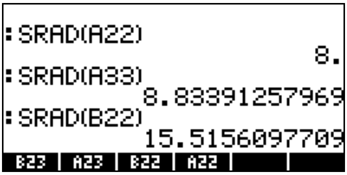
```
                             21
:CNRM(A33)
                             20
:RNRM(A23)
                             19
:CNRM(A23)
                             10
```

**Row norm and column norm of a matrix**

The row norm of a matrix is calculated by taking the sums of the absolute values of all elements in each row, and then, selecting the maximum of these sums. The column norm of a matrix is calculated by taking the sums of the absolute values of all elements in each column, and then, selecting the maximum of these sums.

## Function SRAD

Function SRAD determines the Spectral RADius of a matrix, defined as the largest of the absolute values of its eigenvalues.   For example,



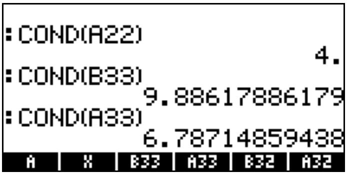
```
:SRAD(A22)
                    8.
:SRAD(A33)
          8.83391257969
:SRAD(B22)
          15.5156097709
 B23 | A23 | B22 | A22 |     |
```

**Definition of eigenvalues and eigenvectors of a matrix**

The eigenvalues of a square matrix result from the matrix equation $\mathbf{A}\cdot\mathbf{x} = \lambda\cdot\mathbf{x}$. The values of $\lambda$ that satisfy the equation are known as the eigenvalues of the matrix **A**.  The values of x that result from the equation for each value of l are known as the eigenvectors of the matrix.   Further details on calculating eigenvalues and eigenvectors are presented later in the chapter.

## Function COND

Function COND determines the condition number of a matrix.  Examples,



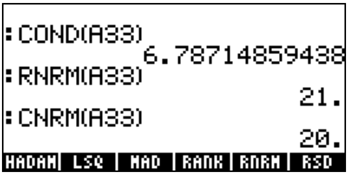
```
:COND(A22)
                    4.
:COND(B33)
          9.88617886179
:COND(A33)
          6.78714859438
  A  |  X  | B33 | A33 | B32 | A32
```

**Condition number of a matrix**

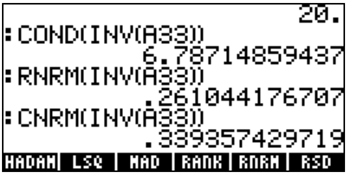
The condition number of a square non-singular matrix is defined as the product of the matrix norm times the norm of its inverse, i.e., cond(**A**) = $||\mathbf{A}||\times||\mathbf{A}^{-1}||$.   We will choose as the matrix norm, $||\mathbf{A}||$, the maximum of its row norm (RNRM) and column norm (CNRM), while the norm of the inverse, $||\mathbf{A}^{-1}||$, will be selected as the minimum of its row norm and column norm.  Thus,  $||\mathbf{A}||$ = max(RNRM(**A**),CNRM(**A**)), and  $||\mathbf{A}^{-1}||$ = min(RNRM($\mathbf{A}^{-1}$), CNRM($\mathbf{A}^{-1}$)).

The condition number of a singular matrix is infinity. The condition number of a non-singular matrix is a measure of how close the matrix is to being singular. The larger the value of the condition number, the closer it is to singularity. (A singular matrix is one for which the inverse does not exist).

Try the following exercise for matrix condition number on matrix A33. The condition number is COND(A33) , row norm, and column norm for A33 are shown to the left. The corresponding numbers for the inverse matrix, INV(A33), are shown to the right:



```
:COND(A33)
                6.78714859438
:RNRM(A33)
                          21.
:CNRM(A33)
                          20.
```

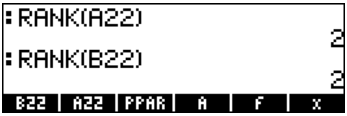


```
                          20.
:COND(INV(A33))
                6.78714859437
:RNRM(INV(A33))
                .261044176707
:CNRM(INV(A33))
                .339357429719
```

Since RNRM(A33) > CNRM(A33), then we take ||A33|| = RNRM(A33) = 21. Also, since CNRM(INV(A33)) < RNRM(INV(A33)), then we take ||INV(A33)|| = CNRM(INV(A33)) = 0.261044... Thus, the condition number is also calculated as CNRM(A33)*CNRM(INV(A33)) = COND(A33) = 6.7871485...

## Function RANK

Function RANK determines the rank of a square matrix. Try the following examples:

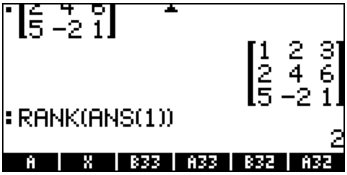


```
:RANK(A22)
                            2
:RANK(B22)
                            2
```

**The rank of a matrix**

The rank of a square matrix is the maximum number of linearly independent rows or columns that the matrix contains. Suppose that you write a square matrix $\mathbf{A}_{n\times n}$ as $\mathbf{A} = [\mathbf{c}_1\ \mathbf{c}_2\ \ldots\ \mathbf{c}_n]$, where $\mathbf{c}_i$ (i = 1, 2, …, n) are vectors representing the columns of the matrix $\mathbf{A}$, then, if any of those columns, say $\mathbf{c}_k$, can be written as $\mathbf{c}_k = \sum_{j\neq k,\, j\in\{1,2,\ldots,n\}} d_j \cdot \mathbf{c}_j$, where the values $d_j$ are constant, we say that $\mathbf{c}_k$ is linearly dependent on the columns included in the summation. (Notice that the values of j include any value in the set {1, 2, …, n}, in any combination, as long as j≠k.) If the expression shown above cannot be written for any of the column vectors then we say that all the columns are linearly independent. A similar definition for the linear independence of rows can be developed by writing the matrix as a column of row vectors. Thus, if we find that rank(**A**) = n, then the matrix has an inverse and it is a non-singular matrix. If, on the other hand, rank(**A**) < n, then the matrix is singular and no inverse exist.

For example, try finding the rank for the matrix:

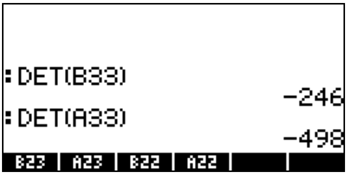


```
 [5 -2 1]
                    [1  2 3]
                    [2  4 6]
                    [5 -2 1]
:RANK(ANS(1))
                          2
```

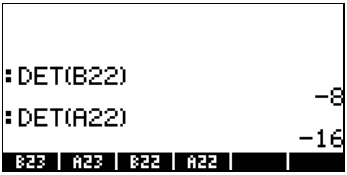
You will find that the rank is 2. That is because the second row [2,4,6] is equal to the first row [1,2,3] multiplied by 2, thus, row two is linearly dependent of row 1 and the maximum number of linearly independent rows is 2. You can check that the maximum number of linearly independent columns is 3. The rank being the maximum number of linearly independent rows or columns becomes 2 for this case.

## Function DET

Function DET calculates the determinant of a square matrix. For example,



```
:DET(B33)
                       -246
:DET(A33)
                       -498
```



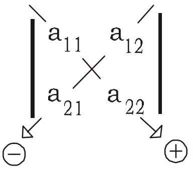
```
:DET(B22)
                         -8
:DET(A22)
                        -16
```

**The determinant of a matrix**

The determinant of  a 2x2 and or a 3x3 matrix are represented by the same arrangement of elements of the matrices, but enclosed between vertical lines, i.e.,

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \qquad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

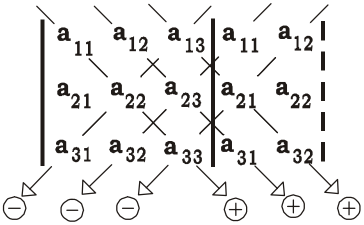
A 2×2 determinant is calculated by multiplying the elements in its diagonal and adding those products accompanied by the positive or negative sign as indicated in the diagram shown below.



The 2×2 determinant is, therefore,

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

A 3×3 determinant is calculated by *augmenting* the determinant, an operation that consists on copying the first two columns of the determinant, and placing them to the right of column 3, as shown in the diagram below.  The diagram also shows the elements to be multiplied with the corresponding sign to attach to their product, in a similar fashion as done earlier for a 2×2 determinant. After multiplication the results are added together to obtain the determinant.



For square matrices of higher order determinants can be calculated by using smaller order determinant called cofactors. The general idea is to “expand” a determinant of a n×n matrix (also referred to as a n×n determinant) into a sum of the cofactors, which are (n-1)×(n-1) determinants, multiplied by the elements of a single row or column, with alternating positive and negative signs. This “expansion” is then carried to the next (lower) level, with cofactors of order (n-2)×(n-2), and so on, until we are left only with a long sum of 2×2 determinants. The 2×2 determinants are then calculated through the method shown above.

The method of calculating a determinant by cofactor expansion is very inefficient in the sense that it involves a number of operations that grows very fast as the size of the determinant increases. A more efficient method, and the one preferred in numerical applications, is to use a result from Gaussian elimination. The method of Gaussian elimination is used to solve systems of linear equations. Details of this method are presented in a later part of this chapter.

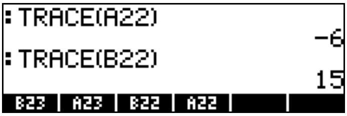
To refer to the determinant of a matrix **A**, we write det(**A**). A singular matrix has a determinant equal to zero.

### Function TRACE

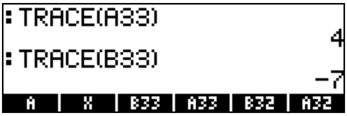
Function TRACE calculates the trace of square matrix, defined as the sum of the elements in its main diagonal, or

$$tr(\mathbf{A}) = \sum_{i=1}^{n} a_{ii} \ .$$

Examples:



```
:TRACE(A22)
                -6
:TRACE(B22)
                15
```



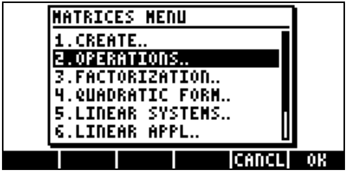
```
:TRACE(A33)
                 4
:TRACE(B33)
                -7
```

### Function TRAN

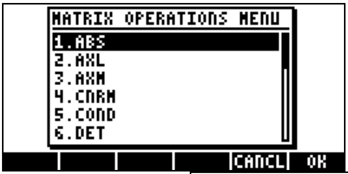
Function TRAN returns the transpose of a real or the conjugate transpose of a complex matrix. TRAN is equivalent to TRN. The operation of function TRN was presented in Chapter 10.

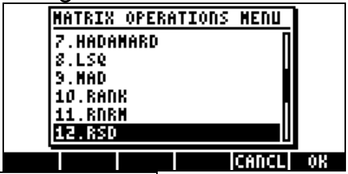
## Additional matrix operations (The matrix OPER menu)

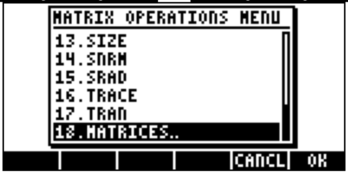
The matrix OPER (OPERATIONS) is available through the keystroke sequence ← MATRICES (system flag 117 set to CHOOSE boxes):



The OPERATIONS menu includes the following functions:



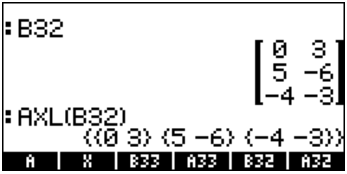




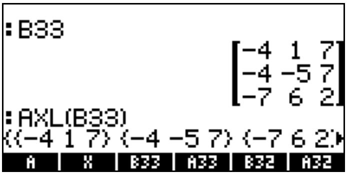
Functions ABS, CNRM, COND, DET, RANK, RNRM, SNRM, TRACE, and TRAN are also found in the MTH/MATRIX/NORM menu (the subject of the previous section). Function SIZE was presented in Chapter 10. Function HADAMARD was presented earlier in the context of matrix multiplication. Functions LSQ , MAD and RSD are related to the solution of systems of linear equations and will be presented in a subsequent section in this Chapter. In this section we'll discuss only functions AXL and AXM

## Function AXL

Function AXL converts an array (matrix) into a list, and vice versa. For examples,



```
:B32
                    [ 0  3]
                    [ 5 -6]
                    [-4 -3]
:AXL(B32)
      {{0 3} {5 -6} {-4 -3}}
 A | X | B33 | A33 | B32 | A32
```

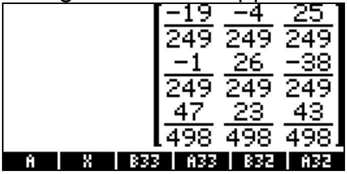


```
:B33
                    [-4  1 7]
                    [-4 -5 7]
                    [-7  6 2]
:AXL(B33)
{{-4 1 7} {-4 -5 7} {-7 6 2▸
 A | X | B33 | A33 | B32 | A32
```

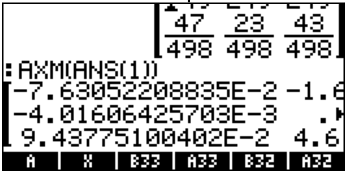
**Note**: the latter operation is similar to that of the program CRMR presented in Chapter 10.

## Function AXM

Function AXM converts an array containing integer or fraction elements into its corresponding decimal, or approximate, form. For example,



```
            [ -19   -4    25 ]
            [ 249  249   249 ]
            [  -1   26   -38 ]
            [ 249  249   249 ]
            [  47   23    43 ]
            [ 498  498   498 ]
 A | X | B33 | A33 | B32 | A32
```

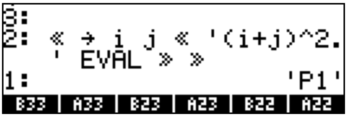


```
            [  47   23    43 ]
            [ 498  498   498 ]
:AXM(ANS(1))
[-7.63052208835E-2 -1.6
[-4.01606425703E-3    .▸
[ 9.43775100402E-2  4.6
 A | X | B33 | A33 | B32 | A32
```

## Function LCXM

Function LCXM can be used to generate matrices such that the element aij is a function of i and j. The input to this function consists of two integers, n and m, representing the number of rows and columns of the matrix to be generated, and a program that takes i and j as input. The numbers n, m, and the program occupy stack levels 3, 2, and 1, respectively. Function LCXM is accessible through the command catalog ↱ _CAT_ .

For example, to generate a 2´3 matrix whose elements are given by $a_{ij}$ = $(i+j)^2$, first, store the following program into variable P1 in RPN mode. This is the way that the RPN stack looks before pressing STO▸.

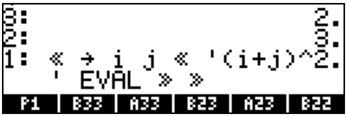


```
3:
2:  « → i j « '(i+j)^2.
    ' EVAL » »
1:                    'P1'
 B33 | A33 | B23 | A23 | B22 | A22
```

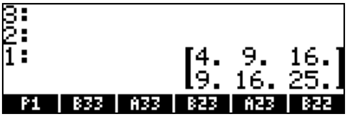
The implementation of function LCXM for  this case requires you to enter:

2 ENTER 3 ENTER ↱ P1 LCXM ENTER

The following figure shows the RPN stack before and after applying function LCXM:

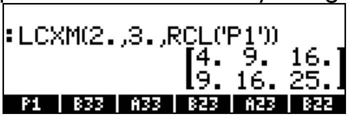


```
3:                    2.
2:                    3.
1:  « → i j « '(i+j)^2.
  ' EVAL » »
 P1 | B33 | A33 | B23 | A23 | B22
```



```
3:
2:
1:              [4.  9.  16.]
                [9. 16.  25.]
 P1 | B33 | A33 | B23 | A23 | B22
```

In ALG mode, this example can be obtained by using:



```
:LCXM(2.,3.,RCL('P1'))
                [4.  9.  16.]
                [9. 16.  25.]
 P1 | B33 | A33 | B23 | A23 | B22
```

The program P1 must still have been created and stored in RPN mode.

## Solution of linear systems

A system of *n* linear equations in *m* variables can be written as

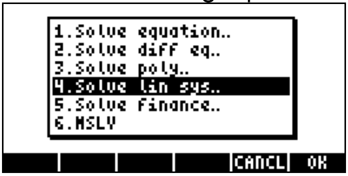
$$
\begin{array}{l}
a_{11}\cdot x_1 + a_{12}\cdot x_2 + a_{13}\cdot x_3 + \ldots + a_{1,m-1}\cdot x_{m-1} + a_{1,m}\cdot x_m = b_1,\\
a_{21}\cdot x_1 + a_{22}\cdot x_2 + a_{23}\cdot x_3 + \ldots + a_{2,m-1}\cdot x_{m-1} + a_{2,m}\cdot x_m = b_2,\\
a_{31}\cdot x_1 + a_{32}\cdot x_2 + a_{33}\cdot x_3 + \ldots + a_{3,m-1}\cdot x_{m-1} + a_{3,m}\cdot x_m = b_3,\\
\quad . \qquad . \qquad . \qquad \ldots \qquad . \qquad . \qquad .\\
\quad . \qquad . \qquad . \qquad \ldots \qquad . \qquad . \qquad .\\
a_{n-1,1}\cdot x_1 + a_{n-1,2}\cdot x_2 + a_{n-1,3}\cdot x_3 + \ldots + a_{n-1,m-1}\cdot x_{m-1} + a_{n-1,m}\cdot x_m = b_{n-1},\\
a_{n1}\cdot x_1 + a_{n2}\cdot x_2 + a_{n3}\cdot x_3 + \ldots + a_{n,m-1}\cdot x_{m-1} + a_{n,m}\cdot x_m = b_n.
\end{array}
$$

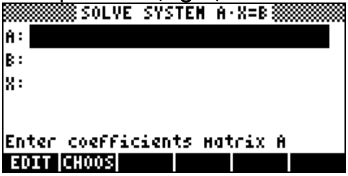
This system of linear equations can be written as a matrix equation, $\mathbf{A}_{n\times m}\cdot\mathbf{x}_{m\times 1} = \mathbf{b}_{n\times 1}$, if we define the following matrix and vectors:

$$
A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}_{n\times m}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}_{m\times 1}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}_{n\times 1}
$$

## Using the numerical solver for linear systems

There are many ways to solve a system of linear equations with the calculator. One possibility is through the numerical solver ↱ NUM.SLV . From the numerical solver screen, shown below (left), select the option *4. Solve lin sys..*, and press OK. The following input form will be provide (right):





Enter coefficients matrix A

To solve the linear system **A**·**x** = **b**, enter the matrix **A**, in the format [[ $a_{11}$, $a_{12,}$ … ], … [….]] in the A: field. Also, enter the vector **b** in the B: field. When the X: field is highlighted, press [SOLVE]. If a solution is available, the solution vector **x** will be shown in the X: field. The solution is also copied to stack level 1. Some examples follow.

### A square system

The system of linear equations

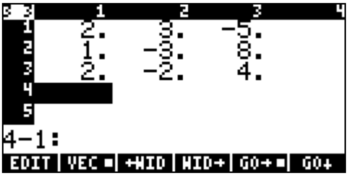
$$2x_1 + 3x_2 -5x_3 = 13,$$
$$x_1 - 3x_2 + 8x_3 = -13,$$
$$2x_1 - 2x_2 + 4x_3 = -6,$$

can be written as the matrix equation **A**·**x** = **b**, if

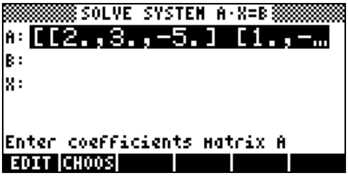
$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \\ 2 & -2 & 4 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad and \quad \mathbf{b} = \begin{bmatrix} 13 \\ -13 \\ -6 \end{bmatrix}.$$

This system has the same number of equations as of unknowns, and will be referred to as a square system. In general, there should be a unique solution to the system. The solution will be the point of intersection of the three planes in the coordinate system ($x_1$, $x_2$, $x_3$) represented by the three equations.

To enter matrix **A** you can activate the Matrix Writer while the A: field is selected. The following screen shows the Matrix Writer used for entering matrix **A**, as well as the input form for the numerical solver after entering matrix **A** (press ENTER in the Matrix Writer):

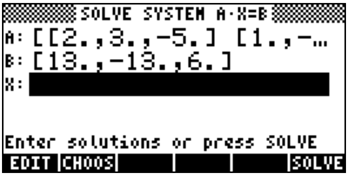


```
     1    2    3    4
1    2.   3.   -5.
2    1.   -3.  8.
3    2.   -2.  4.
4
5
4-1:
```



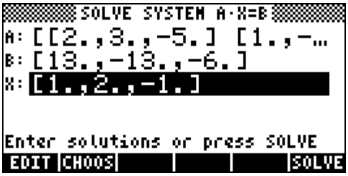
```
SOLVE SYSTEM A·X=B
A: [[2.,3.,-5.] [1.,-…
B:
X:
Enter coefficients matrix A
```

Press ▼ to select the B: field. The vector b can be entered as a row vector with a single set of brackets, i.e., [13,-13,-6] OK .
After entering matrix A and vector b, and with the X: field highlighted, we can press SOLVE to attempt a solution to this system of equations:



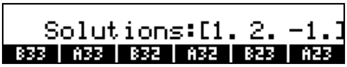
```
SOLVE SYSTEM A·X=B
A: [[2.,3.,-5.] [1.,-…
B: [13.,-13.,6.]
X:
Enter solutions or press SOLVE
```

A solution was found as shown next.



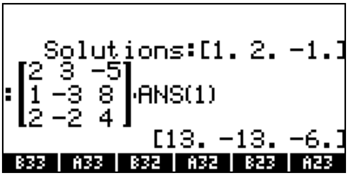
```
SOLVE SYSTEM A·X=B
A: [[2.,3.,-5.] [1.,-…
B: [13.,-13.,-6.]
X: [1.,2.,-1.]
Enter solutions or press SOLVE
```

To see the solution in the stack press ENTER. The solution is **x** = [1,2,-1].



```
Solutions:[1. 2. -1.]
```

To check that the solution is correct, enter the matrix A and multiply times this solution vector (example in algebraic mode):



```
  Solutions:[1. 2. -1.]
 [2  3 -5]
:[1 -3  8]·ANS(1)
 [2 -2  4]
           [13. -13. -6.]
```

### Under-determined system

The system of linear equations

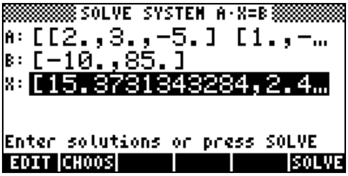
$$2x_1 + 3x_2 –5x_3 = -10,$$
$$x_1 – 3x_2 + 8x_3 = 85,$$

can be written as the matrix equation **A·x** = **b**, if

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad and \quad \mathbf{b} = \begin{bmatrix} -10 \\ 85 \end{bmatrix}.$$

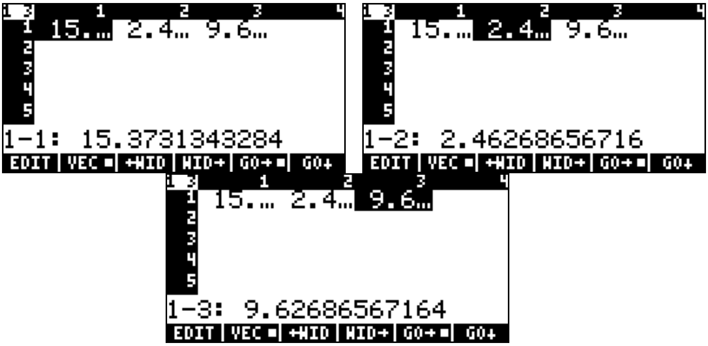
This system has more unknowns than equations, therefore, it is not uniquely determined. We can visualize the meaning of this statement by realizing that each of the linear equations represents a plane in the three-dimensional Cartesian coordinate system ($x_1$, $x_2$, $x_3$). The solution to the system of equations shown above will be the intersection of two planes in space. We know, however, that the intersection of two (non-parallel) planes is a straight line, and not a single point. Therefore, there is more than one point that satisfy the system. In that sense, the system is not uniquely determined.

Let's use the numerical solver to attempt a solution to this system of equations: ↱ NUM.SLV ▼ ▼ ▼ OK . Enter matrix A and vector b as illustrated in the previous example, and press SOLVE when the X: field is highlighted:



```
SOLVE SYSTEM A·X=B
A: [[2.,3.,-5.] [1.,-…
B: [-10.,85.]
X: [15.3731343284,2.4…

Enter solutions or press SOLVE
EDIT CHOOS SOLVE
```

To see the details of the solution vector, if needed, press the EDIT button. This will activate the Matrix Writer. Within this environment, use the right- and left-arrow keys to move about the vector, e.g.,



```
1-1: 15.3731343284
EDIT VEC ■ ←WID WID→ GO→■ GO↓
```

```
1-2: 2.46268656716
EDIT VEC ■ ←WID WID→ GO→■ GO↓
```

```
1-3: 9.62686567164
EDIT VEC ■ ←WID WID→ GO→■ GO↓
```

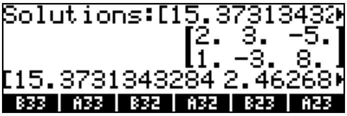
Thus, the solution is **x** = [15.373, 2.4626, 9.6268].

To return to the numerical solver environment, press ENTER.

The procedure that we describe next can be used to copy the matrix A and the solution vector X into the stack. To check that the solution is correct, try the following:

- Press ▲▲, to highlight the A: field.
- Press NXT CALC ENTER, to copy matrix A onto the stack.
- Press OK to return to the numerical solver environment.
- Press ▼ ▼ CALC ENTER, to copy solution vector X onto the stack.
- Press OK to return to the numerical solver environment.
- Press ENTER to return to the stack.

In ALG mode, the stack will now look like this:



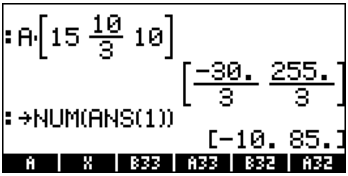
```
Solutions:[15.37313432▸
              [2.  3.  -5.]
              [1. -3.   8. ]
[15.3731343284 2.46268▸
```

Let's store the latest result in a variable X, and the matrix into variable A, as follows:

Press STO▶ ALPHA X ENTER to store the solution vector into variable X
Press ◀ ◀ ◀ to clear three levels of the stack
Press STO▶ ALPHA A ENTER to store the matrix into variable A

Now, let's verify the solution by using: A × X ENTER, which results in (press ▽ to see the vector elements): [-9.99999999999 85. ], close enough to the original vector **b** = [-10 85].

Try also this, A × [15,10/3,10] ENTER ↱ →NUM ENTER, i.e.,



```
:A·[15 10/3 10]
                 [-30./3  255./3]
:→NUM(ANS(1))
                 [-10. 85.]
```

This result indicates that **x** = [15,10/3,10] is also a solution to the system, confirming our observation that a system with more unknowns than equations is not uniquely determined (under-determined).

How does the calculator came up with the solution **x** = [15.37… 2.46… 9.62…] shown earlier? Actually, the calculator minimizes the distance from a point, which will constitute the solution, to each of the planes represented by the equations in the linear system. The calculator uses a *least-square method*, i.e., minimizes the sum of the squares of those distances or errors.

### Over-determined system

The system of linear equations

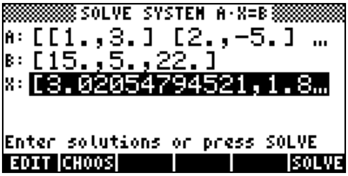
$$x_1 + 3x_2 = 15,$$
$$2x_1 - 5x_2 = 5,$$
$$-x_1 + x_2 = 22,$$

can be written as the matrix equation **A**·**x** = **b**, if

$$\mathbf{A}=\begin{bmatrix}1 & 3\\ 2 & -5\\ -1 & 1\end{bmatrix},\quad \mathbf{x}=\begin{bmatrix}x_1\\ x_2\end{bmatrix},\quad and \quad \mathbf{b}=\begin{bmatrix}15\\ 5\\ 22\end{bmatrix}.$$

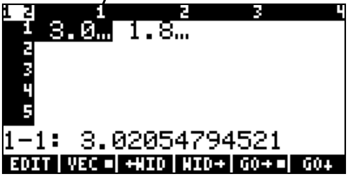
This system has more equations than unknowns (an over-determined system). The system does not have a single solution. Each of the linear equations in the system presented above represents a straight line in a two-dimensional Cartesian coordinate system ($x_1$, $x_2$). Unless two of the three equations in the system represent the same equation, the three lines will have more than one intersection points. For that reason, the solution is not unique. Some numerical algorithms can be used to force a solution to the system by minimizing the distance from the presumptive solution point to each of the lines in the system. Such is the approach followed by the HP 49 G numerical solver.

Let's use the numerical solver to attempt a solution to this system of equations: ↱ NUM.SLV ▼ ▼ ▼ OK . Enter matrix A and vector b as illustrated in the previous example, and press SOLVE when the X: field is highlighted:


SOLVE SYSTEM A·X=B
A: [[1.,3.] [2.,-5.] …
B: [15.,5.,22.]
X: [3.02054794521,1.8…

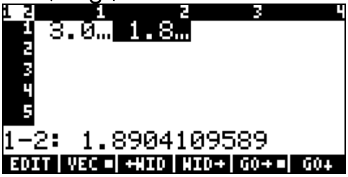
Enter solutions or press SOLVE

To see the details of the solution vector, if needed, press the EDIT button. This will activate the Matrix Writer. Within this environment, use the right- and left-arrow keys to move about the vector, e.g.,



3.0… 1.8…

1-1: 3.02054794521



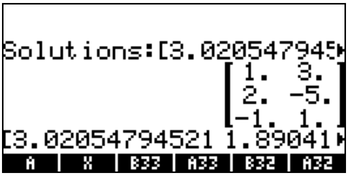
3.0… 1.8…

1-2: 1.8904109589

Press ENTER to return to the numerical solver environment. To check that the solution is correct, try the following:

- Press ▲ ▲, to highlight the A: field.
- Press NXT CALC ENTER, to copy matrix A onto the stack.
- Press OK to return to the numerical solver environment.
- Press ▼ ▼ CALC ENTER, to copy solution vector X onto the stack.
- Press OK to return to the numerical solver environment.
- Press ENTER to return to the stack.

In ALG mode, the stack will now look like this:



```
Solutions:[3.020547945▸
                 [ 1.  3.]
                 [ 2. -5.]
                 [-1.  1.]
[3.02054794521 1.89041▸
```

Let's store the latest result in a variable X, and the matrix into variable A, as follows:

Press STO▸ ALPHA X ENTER to store the solution vector into variable X<br>
Press ◄ ◄ ◄ to clear three levels of the stack<br>
Press STO▸ ALPHA A ENTER to store the matrix into variable A

Now, let's verify the solution by using: A × X ENTER, which results in the vector [8.6917… -3.4109… -1.1301…], which is not equal to [15 5 22], the original vector **b**. The "solution" is simply the point that is closest to the three lines represented by the three equations in the system, and not an exact solution.

## Least-square solution (function LSQ)

The LSQ function returns the minimum-norm least-square solution of a linear system Ax = b, according to the following criteria:

- If **A** is a square matrix and **A** is non-singular (i.e., it's inverse matrix exist, or its determinant is non-zero), LSQ returns the exact solution to the linear system.
- If **A** has less than full row rank (underdetermined system of equations), LSQ returns the solution with the minimum Euclidean length out of an infinity number of solutions.
- If **A** has less than full column rank (over-determined system of equations), LSQ returns the "solution" with the minimum residual value **e** = **A**·**x** – **b**. The system of equations may not have a solution, therefore, the value returned is not a real solution to the system, just the one with the smallest residual.

Function LSQ takes as input vector **b** and matrix **A**, in that order. Function LSQ can be found in Function catalog (↱ CAT). Next, we use function LSQ to repeat the solutions found earlier with the numerical solver:

**Square system**

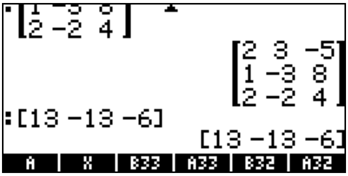
Consider the system

$$2x_1 + 3x_2 -5x_3 = 13,$$
$$x_1 - 3x_2 + 8x_3 = -13,$$
$$2x_1 - 2x_2 + 4x_3 = -6,$$

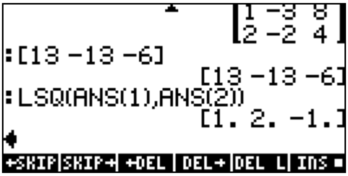
with

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \\ 2 & -2 & 4 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad and \quad \mathbf{b} = \begin{bmatrix} 13 \\ -13 \\ -6 \end{bmatrix}.$$

The solution using LSQ is shown next:



```
 [1 -3 8
 [2 -2 4]
                        [2 3 -5]
                        [1 -3 8]
                        [2 -2 4]
:[13 -13 -6]
                    [13 -13 -6]
```



```
                          [1 -3 8
                          [2 -2 4]
:[13 -13 -6]
                    [13 -13 -6]
:LSQ(ANS(1),ANS(2))
                    [1. 2. -1.]
```

### Under-determined system

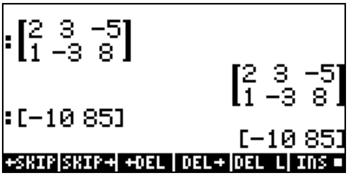
Consider the system

$$2x_1 + 3x_2 - 5x_3 = -10,$$
$$x_1 - 3x_2 + 8x_3 = 85,$$

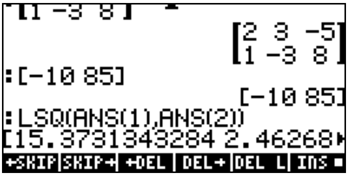
with

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad and \quad \mathbf{b} = \begin{bmatrix} -10 \\ 85 \end{bmatrix}.$$

The solution using LSQ is shown next:



```
:[2 3 -5]
 [1 -3 8]
                [2 3 -5]
                [1 -3 8]
:[-10 85]
                [-10 85]
```



```
 [1 -3 8]
                [2 3 -5]
                [1 -3 8]
:[-10 85]
                [-10 85]
:LSQ(ANS(1),ANS(2))
[15.3731343284 2.46268▸
```

### Over-determined system

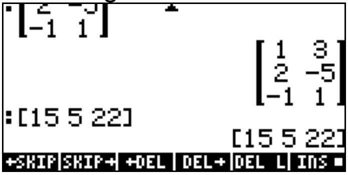
Consider the system

$$x_1 + 3x_2 = 15,$$
$$2x_1 - 5x_2 = 5,$$
$$-x_1 + x_2 = 22,$$

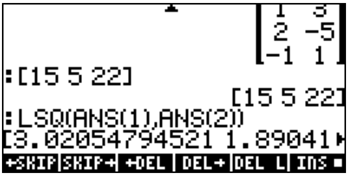
with

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & -5 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad and \quad \mathbf{b} = \begin{bmatrix} 15 \\ 5 \\ 22 \end{bmatrix}.$$

The solution using LSQ is shown next:



```
 [-1 1]
                [1 3]
                [2 -5]
                [-1 1]
:[15 5 22]
                [15 5 22]
```



```
                [1 3]
                [2 -5]
                [-1 1]
:[15 5 22]
                [15 5 22]
:LSQ(ANS(1),ANS(2))
[3.02054794521 1.89041▸
```

Compare these three solutions with the ones calculated with the numerical solver.

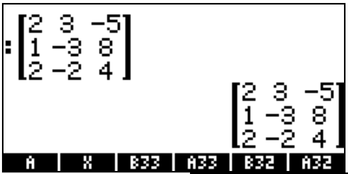
## Solution with the inverse matrix

The solution to the system **A**·**x** = **b**, where **A** is a square matrix is **x** = **A**$^{-1}$· **b**. This results from multiplying the first equation by **A**$^{-1}$, i.e., **A**$^{-1}$·**A**·**x** = **A**$^{-1}$·**b**. By definition, **A**$^{-1}$·**A** = **I**, thus we write **I**·**x** = **A**$^{-1}$·**b**. Also, **I**·**x** = **x**, thus, we have, **x** = **A**$^{-1}$· **b**.

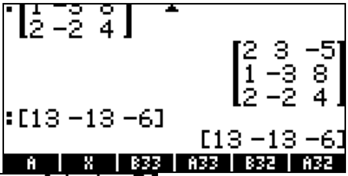
For the example used earlier, namely,

$$2x_1 + 3x_2 –5x_3 = 13,$$
$$x_1 – 3x_2 + 8x_3 = -13,$$
$$2x_1 – 2x_2 + 4x_3 = -6,$$

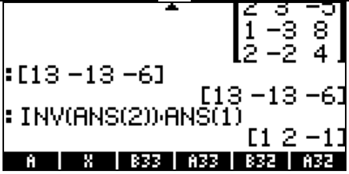
we can find the solution in the calculator as follows:



```
:[2  3 -5]
 [1 -3  8]
 [2 -2  4]
                  [2  3 -5]
                  [1 -3  8]
                  [2 -2  4]
```



```
 [2 -2  4]
                  [2  3 -5]
                  [1 -3  8]
                  [2 -2  4]
:[13 -13 -6]
              [13 -13 -6]
```



```
                  [1 -3  8]
                  [2 -2  4]
:[13 -13 -6]
              [13 -13 -6]
: INV(ANS(2))·ANS(1)
                 [1 2 -1]
```

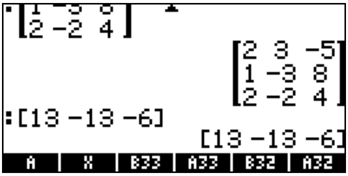
which is the same result found earlier.

## Solution by "division" of matrices

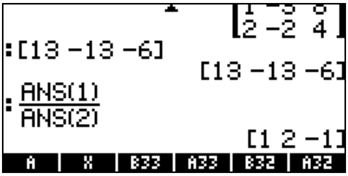
While the operation of division is not defined for matrices, we can use the calculator's ÷ key to "divide" vector **b** by matrix **A** to solve for **x** in the matrix equation **A**·**x** = **b**. This is an arbitrary extension of the algebraic division operation to matrices, i.e., from **A**·**x** = **b**, we dare to write **x** = **b**/**A** (Mathematicians would cringe if they see this!) This, of course is interpreted as (1/**A**)·**b** = **A**$^{-1}$·**b**, which is the same as using the inverse of **A** as in the previous section. The procedure for the case of "dividing" **b** by **A** is illustrated below for the case

$$2x_1 + 3x_2 - 5x_3 = 13,$$
$$x_1 - 3x_2 + 8x_3 = -13,$$
$$2x_1 - 2x_2 + 4x_3 = -6,$$

The procedure is shown in the following screen shots:



```
 [1 -3 8
  2 -2 4]
                    [2  3 -5]
                    [1 -3  8]
                    [2 -2  4]
:[13 -13 -6]
                [13 -13 -6]
```



```
                     [1 -3 8
                      2 -2 4]
:[13 -13 -6]
                [13 -13 -6]
: ANS(1)
  ANS(2)
                   [1 2 -1]
```

The same solution as found above with the inverse matrix.

## Solving multiple set of equations with the same coefficient matrix

Suppose that you want to solve the following three sets of equations:

X +2Y+3Z = 14, 2X +4Y+6Z = 9, 2X +4Y+6Z = -2,
3X -2Y+ Z = 2, 3X -2Y+ Z = -5, 3X -2Y+ Z = 2,
4X +2Y -Z = 5, 4X +2Y -Z = 19, 4X +2Y -Z = 12.

We can write the three systems of equations as a single matrix equation: **A**·**X** = **B**, where

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & -2 & 1 \\ 4 & 2 & -1 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} X_{(1)} & X_{(2)} & X_{(3)} \\ Y_{(1)} & Y_{(2)} & Y_{(3)} \\ Z_{(1)} & Z_{(2)} & Z_{(3)} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} 14 & 9 & -2 \\ 2 & -5 & 2 \\ 5 & 19 & 12 \end{bmatrix}.$$

The sub-indices in the variable names X, Y, and Z, determine to which equation system they refer to. To solve this expanded system we use the following procedure, in RPN mode,

```
[[14,9,-2],[2,-5,2],[5,19,12]] ENTER
[[1,2,3],[3,-2,1],[4,2,-1]] ENTER ÷
```

The result of this operation is:

$$\mathbf{X}=\begin{bmatrix}1 & 2 & 2\\ 2 & 5 & 1\\ 3 & -1 & -2\end{bmatrix}.$$

## Gaussian and Gauss-Jordan elimination

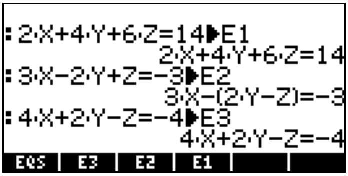
Gaussian elimination is a procedure by which the square matrix of coefficients belonging to a system of n linear equations in n unknowns is reduced to an upper-triangular matrix (*echelon form*) through a series of row operations. This procedure is known as *forward elimination*. The reduction of the coefficient matrix to an upper-triangular form allows for the solution of all *n* unknowns, utilizing only one equation at a time, in a procedure known as *backward substitution*.

### Example of Gaussian elimination using equations

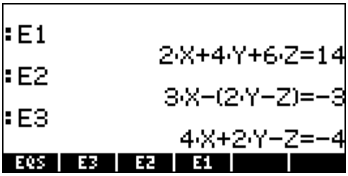
To illustrate the Gaussian elimination procedure we will use the following system of 3 equations in 3 unknowns:

$$\begin{aligned} 2X +4Y+6Z &= 14,\\ 3X -2Y+ Z &= -3,\\ 4X +2Y -Z &= -4. \end{aligned}$$

We can store these equations in the calculator in variables E1, E2, and E3, respectively, as shown below. For backup purposes, a list containing the three equations was also created and stored into variable EQS. This way, if a mistake is made, the equations will still be available to the user.

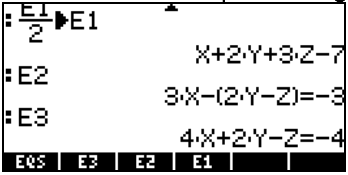


```
:2·X+4·Y+6·Z=14▶E1
          2·X+4·Y+6·Z=14
:3·X-2·Y+Z=-3▶E2
           3·X-(2·Y-Z)=-3
:4·X+2·Y-Z=-4▶E3
            4·X+2·Y-Z=-4
EQS | E3 | E2 | E1 |  |
```



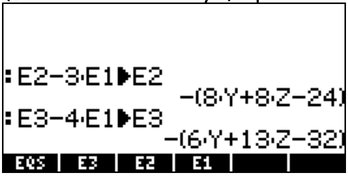
```
:E1
          2·X+4·Y+6·Z=14
:E2
           3·X-(2·Y-Z)=-3
:E3
            4·X+2·Y-Z=-4
EQS | E3 | E2 | E1 |  |
```

To start the process of forward elimination, we divide the first equation (E1) by 2, and store it in E1, and show the three equations again to produce:

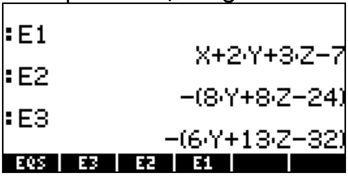


```
:E1/2▶E1
            X+2·Y+3·Z-7
:E2
         3·X-(2·Y-Z)=-3
:E3
          4·X+2·Y-Z=-4
EQS | E3 | E2 | E1 |  |
```

Next, we replace the second equation E2 by (equation 2 – 3×equation 1, i.e., E1-3×E2), and the third by (equation 3 – 4×equation 1), to get

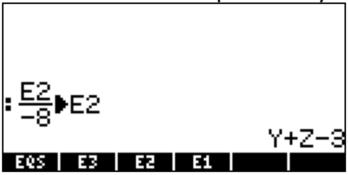


```
:E2-3·E1▶E2
             -(8·Y+8·Z-24)
:E3-4·E1▶E3
           -(6·Y+13·Z-32)
EQS | E3 | E2 | E1 |  |
```

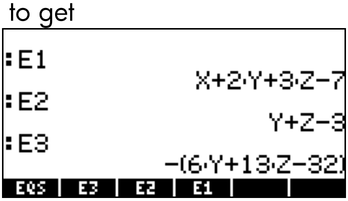


```
:E1
              X+2·Y+3·Z-7
:E2
            -(8·Y+8·Z-24)
:E3
           -(6·Y+13·Z-32)
EQS | E3 | E2 | E1 |  |
```

Next, divide the second equation by –8, to get

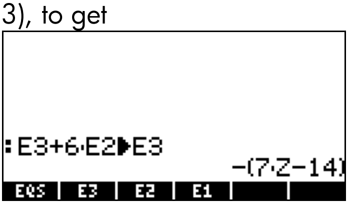


```
:E2/-8▶E2
                    Y+Z-3
EQS | E3 | E2 | E1 |  |
```

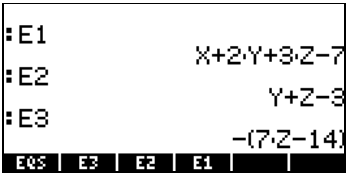


```
:E1
              X+2·Y+3·Z-7
:E2
                    Y+Z-3
:E3
           -(6·Y+13·Z-32)
EQS | E3 | E2 | E1 |  |
```

Next, replace the third equation, E3, with (equation 3 + 6×equation 2, i.e., E2+6×E3), to get



```
:E3+6·E2▶E3
               -(7·Z-14)
EQS | E3 | E2 | E1 |  |
```



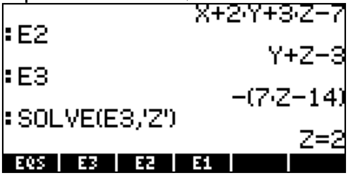
```
:E1
              X+2·Y+3·Z-7
:E2
                    Y+Z-3
:E3
                -(7·Z-14)
EQS | E3 | E2 | E1 |  |
```

Notice that when we perform a linear combination of equations the calculator modifies the result to an expression on the left-hand side of the equal sign, i.e., an expression = 0. Thus, the last set of equations is interpreted to be the following equivalent set of equations:

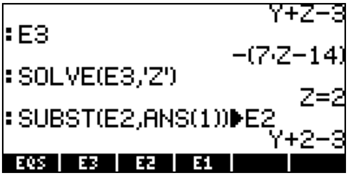
$$X + 2Y + 3Z = 7,$$

$$Y + Z = 3,$$
$$-7Z = -14.$$

The process of backward substitution in Gaussian elimination consists in finding the values of the unknowns, starting from the last equation and working upwards. Thus, we solve for Z first:

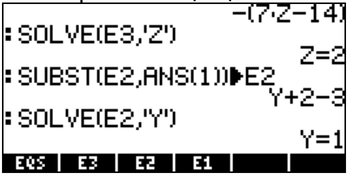


```
              X+2·Y+3·Z-7
:E2
                    Y+Z-3
:E3
                -(7·Z-14)
:SOLVE(E3,'Z')
                      Z=2
```



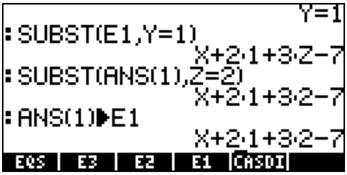
```
                    Y+Z-3
:E3
                -(7·Z-14)
:SOLVE(E3,'Z')
                      Z=2
:SUBST(E2,ANS(1))▶E2
                    Y+2-3
```

Next, we substitute Z=2 into equation 2 (E2), and solve E2 for Y:

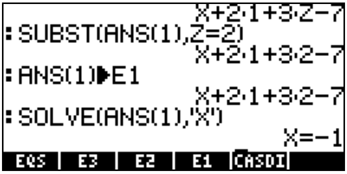


```
                -(7·Z-14)
:SOLVE(E3,'Z')
                      Z=2
:SUBST(E2,ANS(1))▶E2
                    Y+2-3
:SOLVE(E2,'Y')
                      Y=1
```

Next, we substitute Z=2 and Y = 1 into E1, and solve E1 for X:



```
                      Y=1
:SUBST(E1,Y=1)
              X+2·1+3·Z-7
:SUBST(ANS(1),Z=2)
              X+2·1+3·2-7
:ANS(1)▶E1
              X+2·1+3·2-7
```



```
              X+2·1+3·Z-7
:SUBST(ANS(1),Z=2)
              X+2·1+3·2-7
:ANS(1)▶E1
              X+2·1+3·2-7
:SOLVE(ANS(1),'X')
                     X=-1
```

The solution is, therefore, X = -1, Y = 1, Z = 2.

### Example of Gaussian elimination using matrices

The system of equations used in the example above can be written as a matrix equation **A**·**x** = **b**, if we use:

$$\mathbf{A} = \begin{pmatrix} 2 & 4 & 6 \\ 3 & -2 & 1 \\ 4 & 2 & -1 \end{pmatrix}, \quad \mathbf{x} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 14 \\ -3 \\ -4 \end{bmatrix}.$$

To obtain a solution to the system matrix equation using Gaussian elimination, we first create what is known as the *augmented matrix* corresponding to **A**, i.e.,

$$\mathbf{A}_{aug} = \left(\begin{array}{ccc|c} 2 & 4 & 6 & 14 \\ 3 & -2 & 1 & -3 \\ 4 & 2 & -1 & -4 \end{array}\right)$$

The matrix $\mathbf{A}_{aug}$ is the same as the original matrix A with a new row, corresponding to the elements of the vector **b**, added (i.e., augmented) to the right of the rightmost column of **A**.

Once the augmented matrix is put together, we can proceed to perform row operations on it that will reduce the original A matrix into an upper-triangular matrix. For this exercise, we will use the RPN mode (MODE +/- OK), with system flag 117 set to SOFT menu. In your calculator, use the following keystrokes. First, enter the augmented matrix, and make an extra copy of the same in the stack (This step is not necessary, except as an insurance that you have an extra copy of the augmented matrix saved in case you make a mistake in the forward elimination procedure that we are about to undertake.):

[[2,4,6,14],[3,-2,1,-3],[4,2,-1,-4]] ENTER ENTER

Save augmented matrix in variable AAUG: ' ALPHA ALPHA A A U G ALPHA STO▸

With a copy of the augmented matrix in the stack, press ↰ MTH MATRX ROW to activate the ROW operation menu. Next, perform the following row operations on your augmented matrix.
Multiply row 1 by ½: 2 1/x 1 RCI

Multiply row 1 by –3 add it to row 2, replacing it: 3 +/- SPC 1 SPC 2 RCIJ

Multiply row 1 by –4 add it to row 3, replacing it: 4 +/- SPC 1 SPC 3 RCIJ

Multiply row 2 by –1/8: 8 +/- 1/x 2 RCI
Multiply row 2 by 6 add it to row 3, replacing it: 6 SPC 2 SPC 3 RCIJ

If you were performing these operations by hand, you would write the following:

$$\mathbf{A}_{aug} = \left(\begin{array}{ccc|c} 2 & 4 & 6 & 14 \\ 3 & -2 & 1 & -3 \\ 4 & 2 & -1 & -4 \end{array}\right) \cong \left(\begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 3 & -2 & 1 & -3 \\ 4 & 2 & -1 & -4 \end{array}\right)$$

$$\mathbf{A}_{aug} \cong \left(\begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & -8 & -8 & -24 \\ 0 & -6 & -13 & -32 \end{array}\right) \cong \left(\begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 3 \\ 0 & -6 & -13 & -32 \end{array}\right)$$

$$\mathbf{A}_{aug} \cong \left(\begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & -7 & -14 \end{array}\right)$$

The symbol ≅ (" is equivalent to") indicates that what follows is equivalent to the previous matrix with some row (or column) operations involved.

The resulting matrix is upper-triangular, and equivalent to the set of equations

$$X + 2Y + 3Z = 7,$$
$$Y + Z = 3,$$
$$-7Z = -14,$$

which can now be solved, one equation at a time, by backward substitution, as in the previous example.

### Gauss-Jordan elimination using matrices

Gauss-Jordan elimination consists in continuing the row operations in the upper-triangular matrix resulting from the forward elimination process until an identity matrix results in place of the original **A** matrix. For example, for the case we just presented, we can continue the row operations as follows:

Multiply row 3 by –1/7: 7 +/– 1/x 3 RCI

Multiply row 3 by –1, add it to row 2, replacing it: 1 +/– SPC 3 SPC 2 RCIJ

Multiply row 3 by –3, add it to row 1, replacing it:
3 +/- SPC 3 SPC 1 RCIJ

Multiply row 2 by –2, add it to row 1, replacing it: 2 +/- SPC 2 SPC 1 RCIJ

Writing this process by hand will result in the following steps:

$$\mathbf{A}_{aug} = \left(\begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & -7 & -14 \end{array}\right) \cong \left(\begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & 1 & 2 \end{array}\right) \cong \left(\begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 \end{array}\right)$$

$$A_{aug} \cong \left(\begin{array}{ccc|c} 1 & 2 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array}\right) \cong \left(\begin{array}{ccc|c} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array}\right).$$

**Pivoting**

If you look carefully at the row operations in the examples shown above, you will notice that many of those operations divide a row by its corresponding element in the main diagonal. This element is called a pivot element, or simply, a *pivot*. In many situations it is possible that the pivot element become zero, in which case we cannot divide the row by its pivot. Also, to improve the numerical solution of a system of equations using Gaussian or Gauss-Jordan elimination, it is recommended that the pivot be the element with the largest absolute value in a given column. In such cases, we exchange rows before performing row operations. This exchange of rows is called *partial pivoting*. To follow this recommendation is it often necessary to exchange rows in the augmented matrix while performing a Gaussian or Gauss-Jordan elimination.

While performing pivoting in a matrix elimination procedure, you can improve the numerical solution even more by selecting as the pivot the element with the largest absolute value in the column and row of interest. This operation may require exchanging not only rows, but also columns, in some pivoting operations. When row and column exchanges are allowed in pivoting, the procedure is known as <u>*full pivoting*</u>.

When exchanging rows and columns in partial or full pivoting, it is necessary to keep track of the exchanges because the order of the unknowns in the solution is altered by those exchanges. One way to keep track of column exchanges in partial or full pivoting mode, is to create a <u>*permutation matrix*</u> **P** = $\mathbf{I}_{n\times n}$, at the beginning of the procedure. Any row or column exchange required in the augmented matrix $\mathbf{A}_{aug}$ is also registered as a row or column exchange, respectively, in the permutation matrix. When the solution is achieved, then, we multiply the permutation matrix by the unknown vector **x** to obtain the order of the unknowns in the solution. In other words, the final solution is given by $\mathbf{P}\cdot\mathbf{x} = \mathbf{b}'$, where $\mathbf{b}'$ is the last column of the augmented matrix after the solution has been found.

**Example of Gauss-Jordan elimination with full pivoting**

Let's illustrate full pivoting with an example. Solve the following system of equations using full pivoting and the Gauss-Jordan elimination procedure:

$$X + 2Y + 3Z = 2,$$
$$2X + \quad 3Z = -1,$$
$$8X + 16Y - Z = 41.$$

The augmented matrix and the permutation matrix are as follows:

$$\mathbf{A}_{aug} = \begin{bmatrix} 1 & 2 & 3 & 2 \\ 2 & 0 & 3 & -1 \\ 8 & 16 & -1 & 41 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Store the augmented matrix in variable AAUG, then press ↱ AAUG to get a copy in the stack. We want to keep the CSWP (Column Swap) command readily available, for which we use: ↱ _CAT_ ALPHA ALPHA C S ALPHA (find CSWP), OK. You'll get an error message, press ON, and ignore the message.
Next, get the ROW menu available by pressing: ↰ _MATRICES_ CREAT ROW.

Now we are ready to start the Gauss-Jordan elimination with full pivoting. We will need to keep track of the permutation matrix by hand, so take your notebook and write the **P** matrix shown above.

First, we check the pivot $a_{11}$. We notice that the element with the largest absolute value in the first row and first column is the value of $a_{31}$ = 8. Since we want this number to be the pivot, then we exchange rows 1 and 3, by using: 1 SPC 3 NXT RSWP. The augmented matrix and the permutation matrix now are:

$$\begin{bmatrix} 8 & 16 & -1 & 41 \\ 2 & 0 & 3 & -1 \\ 1 & 2 & 3 & 2 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Checking the pivot at position (1,1) we now find that 16 is a better pivot than 8, thus, we perform a column swap as follows: 1 SPC 2 ↱ CAT OK. CSWP. The augmented matrix and the permutation matrix now are:

$$\begin{bmatrix} 16 & 8 & -1 & 41 \\ 0 & 2 & 3 & -1 \\ 2 & 1 & 3 & 2 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Now we have the largest possible value in position (1,1), i.e., we performed full pivoting at (1,1). Next, we proceed to divide by the pivot:
1 6 1/x 1 NXT RCI . The permutation matrix does not change, but the augmented matrix is now:

$$\begin{bmatrix} 1 & 1/2 & -1/16 & 41/16 \\ 0 & 2 & 3 & -1 \\ 2 & 1 & 3 & 2 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The next step is to eliminate the 2 from position (3,2) by using:
2 +/- SPC 1 SPC 3 RCIJ

$$\begin{bmatrix} 1 & 1/2 & -1/16 & 41/16 \\ 0 & 2 & 3 & -1 \\ 0 & 0 & 25/8 & -25/8 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Having filled up with zeros the elements of column 1 below the pivot, now we proceed to check the pivot at position (2,2). We find that the number 3 in position (2,3) will be a better pivot, thus, we exchange columns 2 and 3 by using: 2 SPC 3 ↱ CAT OK

$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

$$\begin{matrix} 0 & 3 & 2 & -1 \\ 0 & 25/8 & 0 & -25/82 \end{matrix}\rfloor \qquad \lfloor\begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}\rfloor$$

Checking the pivot at position (2,2), we now find that the value of 25/8, at position (3,2), is larger than 3. Thus, we exchange rows 2 and 3 by using: 2 SPC 3 NXT RSWP

$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 25/8 & 0 & -25/8 \\ 0 & 3 & 2 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Now, we are ready to divide row 2 by the pivot 25/8, by using ' 8 ÷ 2 5 ► SPC 2 NXT RCI

$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 1 & 0 & -1 \\ 0 & 3 & 2 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Next, we eliminate the 3 from position (3,2) by using: 3 +/- SPC 2 SPC 3 RCIJ

$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 2 & 2 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Having filled with zeroes the position below the pivot, we proceed to check the pivot at position (3,3). The current value of 2 is larger than ½ or 0, thus, we keep it unchanged. We do divide the whole third row by 2 to convert the pivot to 1, by using: 2 1/x 3 RCI

$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Next, we proceed to eliminate the ½ in position (1,3) by using: 2 1/x +/- SPC 3 SPC 1 RCIJ

$$\begin{bmatrix} 1 & -1/16 & 0 & 33/16 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Finally, we eliminate the –1/16 from position (1,2) by using:
1 6 1/x SPC 2 SPC 1 RCIJ

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

We now have an identity matrix in the portion of the augmented matrix corresponding to the original coefficient matrix A, thus we can proceed to obtain the solution while accounting for the row and column exchanges coded in the permutation matrix **P**. We identify the unknown vector **x**, the modified independent vector **b**′ and the permutation matrix **P** as:

$$\mathbf{x} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \mathbf{b'} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

The solution is given by **P**·**x**=**b**′, or

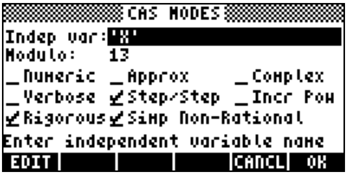
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix}.$$

Which results in

$$\begin{bmatrix} Y \\ Z \\ X \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix}.$$

## Step-by-step calculator procedure for solving linear systems

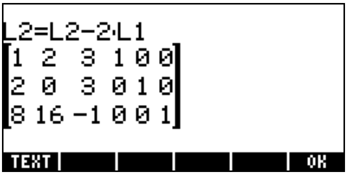
The example we just worked is, of course, the step-by-step, user-driven procedure to use full pivoting for Gauss-Jordan elimination solution of linear equation systems. You can see the step-by-step procedure used by the calculator to solve a system of equations, without user intervention, by setting the step-by-step option in the calculator's CAS, as follows:



```
CAS MODES
Indep var:'X'
Modulo:   13
_Numeric  _Approx    _Complex
_Verbose  ✓Step/Step _Incr Pow
✓Rigorous ✓Simp Non-Rational
Enter independent variable name
```

Then, for this particular example, in RPN mode, use:
`[2,-1,41]` ENTER `[[1,2,3],[2,0,3],[8,16,-1]]` ENTER ÷

The calculator shows an augmented matrix consisting of the coefficients matrix **A** and the identity matrix **I**, while, at the same time, showing the next procedure to calculate:

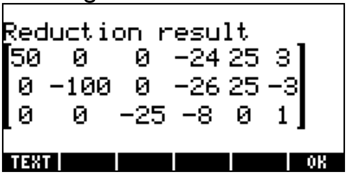


```
L2=L2-2·L1
[1  2  3 1 0 0]
[2  0  3 0 1 0]
[8 16 -1 0 0 1]
```

L2 = L2-2·L1 stands for "replace row 2 (L2) with the operation L2 – 2·L1. If we had done this operation by hand, it would have corresponded to: 2 +/- SPC 1 SPC 1 RCI. Press OK, and follow the operations in your calculator's screen. You will see the following operations performed:

L3=L3-8·L1, L1 = 2·L1--1·L2, L1=25·L1--3·L3, L2 = 25·L2-3·L3,

and finally a message indicating "Reduction result" showing:



```
Reduction result
[50    0    0  -24 25  3]
[ 0 -100    0  -26 25 -3]
[ 0    0  -25   -8  0  1]
```

When you press OK , the calculator returns the final result [1 2 –1].

### Calculating the inverse matrix step-by-step

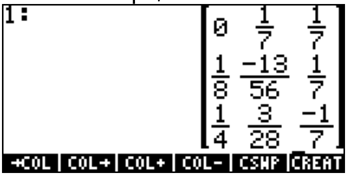
The calculation of an inverse matrix can be considered as calculating the solution to the augmented system [**A** | **I** ]. For example, for the matrix **A** used in the previous example, we would write this augmented matrix as

$$\mathbf{A}_{aug(I)} = \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 3 & -2 & 1 & 0 & 1 & 0 \\ 4 & 2 & -1 & 0 & 0 & 1 \end{array}\right].$$

To see the intermediate steps in calculating and inverse, just enter the matrix **A** from above, and press 1/x, while keeping the step-by-step option active in the calculator's CAS. Use the following:

```
[[ 1,2,3],[3,-2,1],[4,2,-1]] ENTER 1/x
```

After going through the different steps, the solution returned is:



1:

$$\begin{bmatrix} 0 & \frac{1}{7} & \frac{1}{7} \\ \frac{1}{8} & \frac{-13}{56} & \frac{1}{7} \\ \frac{1}{4} & \frac{3}{28} & \frac{-1}{7} \end{bmatrix}$$

What the calculator showed was not exactly a Gauss-Jordan elimination with full pivoting, but a way to calculate the inverse of a matrix by performing a Gauss-Jordan elimination, without pivoting. This procedure for calculating the inverse is based on the augmented matrix $(\mathbf{A}_{aug})_{n\times n} = [\mathbf{A}_{n\times n} \mid \mathbf{I}_{n\times n}]$.

The calculator showed you the steps up to the point in which the left-hand half of the augmented matrix has been converted to a diagonal matrix. From there, the final step is to divide each row by the corresponding main diagonal pivot. In other words, the calculator has transformed $(\mathbf{A}_{aug})_{n\times n} = [\mathbf{A}_{n\times n} \mid \mathbf{I}_{n\times n}]$, into $[\mathbf{I} \mid \mathbf{A}^{-1}]$.

### Inverse matrices and determinants

Notice that all the elements in the inverse matrix calculated above are divided by the value 56 or one of its factors (28, 7, 8, 4 or 1). If you calculate the determinant of the matrix **A**, you get *det*(**A**) = 56.

We could write, $\mathbf{A}^{-1} = \mathbf{C}/det(\mathbf{A})$, where **C** is the matrix

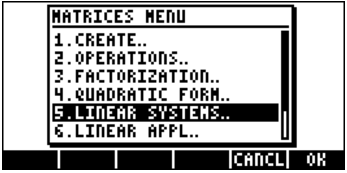
$$\mathbf{C} = \begin{bmatrix} 0 & 8 & 8 \\ 7 & -13 & 8 \\ 14 & 6 & -8 \end{bmatrix}.$$

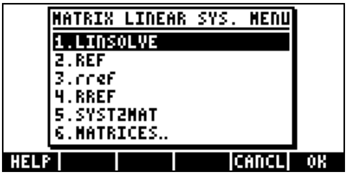
The result $(\mathbf{A}^{-1})_{n\times n} = \mathbf{C}_{n\times n} / det(\mathbf{A}_{n\times n})$, is a general result that applies to any non-singular matrix **A**. A general form for the elements of C can be written based on the Gauss-Jordan algorithm.

Based on the equation $\mathbf{A}^{-1} = \mathbf{C}/\det(\mathbf{A})$, sketched above, the inverse matrix, $\mathbf{A}^{-1}$, is not defined if *det*(**A**) = 0. Thus, the condition *det*(**A**) = 0 defines also a singular matrix.

## Solution to linear systems using calculator functions

The simplest way to solve a system of linear equations, $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$, in the calculator is to enter **b**, enter **A**, and then use the division function /. If the system of linear equations is over-determined or under-determined, a "solution" can be produced by using Function LSQ (Least-SQuares), as shown earlier. The calculator, however, offers other possibilities for solving linear systems of equations by using Functions included in the MATRICES' LINEAR SYSTEMS.. menu accessible through ⇦ MATRICES (Set system flag 117 to CHOOSE boxes):

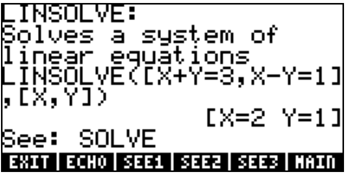




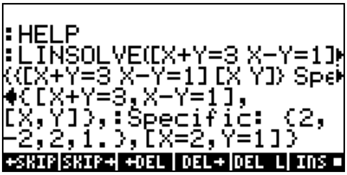
The functions included are LINSOLVE, REF, rref, RREF, and SYST2MAT.

### Function LINSOLVE

Function LINSOLVE takes as arguments an array of equations and a vector containing the names of the unknowns, and produces the solution to the linear system. The following screens show the help-facility entry (see Chapter 1) for function LINSOLVE, and the corresponding example listed in the entry. The right-hand side screen shows the result using the line editor (press ▼ to activate):



```
LINSOLVE:
Solves a system of
linear equations
LINSOLVE([X+Y=3,X-Y=1]
,[X,Y])
                [X=2 Y=1]
See: SOLVE
```



```
:HELP
:LINSOLVE([X+Y=3 X-Y=1]▸
{([X+Y=3 X-Y=1] [X Y]} Spe▸
◂{[X+Y=3,X-Y=1],
[X,Y]},:Specific: {2,
-2,2,1.},[X=2,Y=1]}
```

Here is another example in ALG mode. Enter the following:

```
LINSOLVE([X-2*Y+Z=-8,2*X+Y-2*Z=6,5*X-2*Y+Z=-12],
                    [X,Y,Z])
```

to produce the solution: [X=-1,Y=2,Z = -3].

Function LINSOLVE works with symbolic expressions. Functions REF, rref, and RREF, work with the augmented matrix in a Gaussian elimination approach.

### Functions REF, rref, RREF

The upper triangular form to which the augmented matrix is reduced during the forward elimination part of a Gaussian elimination procedure is known as an "echelon" form. Function REF (Reduce to Echelon Form) produces such a matrix given the augmented matrix in stack level 1.

Consider the augmented matrix,

$$\mathbf{A}_{aug} = \left[\begin{array}{ccc|c} 1 & -2 & 1 & 0 \\ 2 & 1 & -2 & -3 \\ 5 & -2 & 1 & 12 \end{array}\right].$$

Representing a linear system of equations, **A**·**x** = **b**, where

**A** = [[1,-2,1],[2,1,-2],[5,-2,1]],

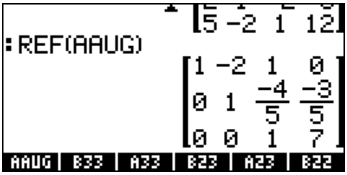
and

**b** = [[0],[-3],[12]].

Enter the augmented matrix, and save it into variable AAUG, in ALG mode:

```
[[1,-2,1,0],[2,1,-2,-3][5,-2,1,12]] ▶AAUG
```

Application of function REF produces:



[5 -2 1 12]

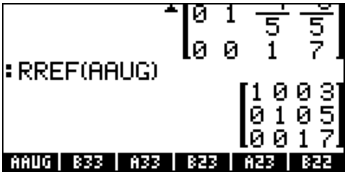
:REF(AAUG)

$$\begin{bmatrix} 1 & -2 & 1 & 0 \\ 0 & 1 & \frac{-4}{5} & \frac{-3}{5} \\ 0 & 0 & 1 & 7 \end{bmatrix}$$

The result is the upper triangular (echelon form) matrix of coefficients resulting from the forward elimination step in a Gaussian elimination procedure.

The diagonal matrix that results from a Gauss-Jordan elimination is called a row-reduced echelon form. Function RREF ( Row-Reduced Echelon Form) The results of this function call is to produce the row-reduced echelon form so that the matrix of coefficients is reduced to an identity matrix. The extra column in the augmented matrix will contain the solution to the system of equations.

As an example, we show the result of applying function RREF to matrix AAUG in ALG mode:

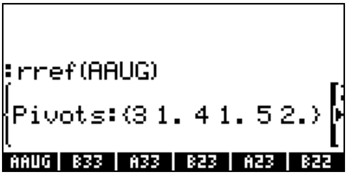


```
:RREF(AAUG)
                    [1 0 0 3]
                    [0 1 0 5]
                    [0 0 1 7]
```

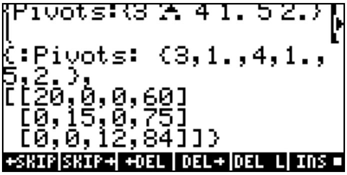
The result is final augmented matrix resulting from a Gauss-Jordan elimination without pivoting.

A *r*ow-*r*educed *e*chelon *f*orm for an augmented matrix can be obtained by using function *rref*. This function produces a list of the pivots and an equivalent matrix in row-reduced echelon form so that the matrix of coefficients is reduced to a diagonal matrix.

For example, for matrix AAUG, function rref produces the following result:



```
:rref(AAUG)
{Pivots:{3 1. 4 1. 5 2.}
```

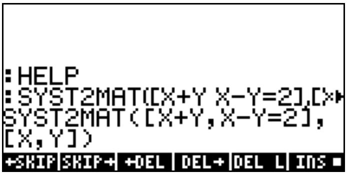


```
{:Pivots: {3,1.,4,1.,
5,2.},
[[20,0,0,60]
 [0,15,0,75]
 [0,0,12,84]]}
```

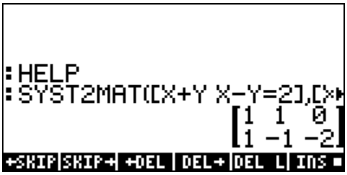
The second screen above is obtained by activating the line editor (press ▼). The result shows pivots of 3, 1, 4, 1, 5, and 2, and a reduced diagonal matrix.

**Function SYST2MAT**

This function converts a system of linear equations into its augmented matrix equivalent. The following example is available in the help facility of the calculator:



```
:HELP
:SYST2MAT([X+Y X-Y=2],[X▸
SYST2MAT([X+Y,X-Y=2],
[X,Y])
```



```
:HELP
:SYST2MAT([X+Y X-Y=2],[X▸
```

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & -2 \end{bmatrix}$$

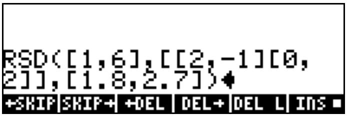
The result is the augmented matrix corresponding to the system of equations:

$$X+Y = 0$$
$$X\text{-}Y =2$$

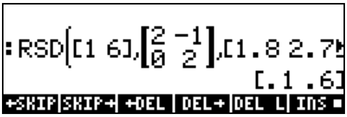
## Residual errors in linear system solutions (Function RSD)

Function RSD calculates the ReSiDuals or errors in the solution of the matrix equation $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$, representing a system of n linear equations in n unknowns. We can think of solving this system as solving the matrix equation: $f(\mathbf{x}) = \mathbf{b} - \mathbf{A}\cdot\mathbf{x} = 0$. Suppose that, through a numerical method, we produce as a first approximation the solution $\mathbf{x}(0)$. Evaluating $f(\mathbf{x}(0)) = \mathbf{b} - \mathbf{A}\cdot\mathbf{x}(0) = \mathbf{e} \neq 0$. Thus, **e** is a vector of residuals of Function for the vector $\mathbf{x} = \mathbf{x}\ (0)$.

To use Function RSD you need the terms **b**, **A**, and **x**(0), as arguments. The vector returned is $\mathbf{e} = \mathbf{b} - \mathbf{A}\cdot\mathbf{x}(0)$. For example, using **A** = `[[2,-1][0,2]]`, **x**(0) = `[1.8,2.7]`, and **b** = `[1,6]`, we can find the vector of residuals as follows:



```
RSD([1,6],[[2,-1][0,
2]],[1.8,2.7])◂
```



```
:RSD([1 6],[[2 -1][0 2]],[1.8 2.7▸
                          [.1 .6]
```

The result is $\mathbf{e} = \mathbf{b} - \mathbf{A}\cdot\mathbf{x}(0)$ = `[ 0.1 0.6 ].`

**Note**: If we let the vector $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}\ (0)$, represent the correction in the values of **x** (0), we can write a new matrix equation for $\Delta\mathbf{x}$, namely $\mathbf{A}\cdot\Delta\mathbf{x} = \mathbf{e}$. Solving for $\Delta\mathbf{x}$ we can find the actual solution of the original system as $\mathbf{x} = \mathbf{x}(0) + \Delta\mathbf{x}$.

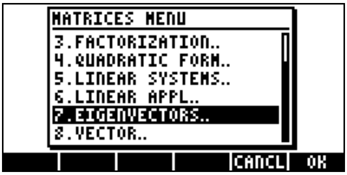
## Eigenvalues and eigenvectors

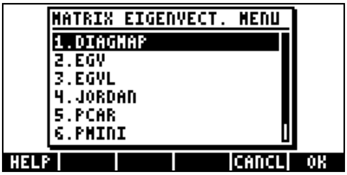
Given a square matrix **A**, we can write the eigenvalue equation **A**·**x** = λ·**x**, where the values of λ that satisfy the equation are known as the eigenvalues of matrix **A**. For each value of λ, we can find, from the same equation, values of **x** that satisfy the eigenvalue equation. These values of **x** are known as the eigenvectors of matrix **A**. The eigenvalues equation can be written also as (**A** – λ·**I**)**x** = 0.

This equation will have a non-trivial solution only if the matrix (**A** – λ·**I**) is singular, i.e., if det(**A** – λ·**I**) = 0.

The last equation generates an algebraic equation involving a polynomial of order *n* for a square matrix $\mathbf{A}_{n\times n}$. The resulting equation is known as the characteristic polynomial of matrix **A**. Solving the characteristic polynomial produces the eigenvalues of the matrix.

The calculator provides a number of functions that provide information regarding the eigenvalues and eigenvectors of a square matrix. Some of these functions are located under the menu MATRICES/EIGEN activated through ↰ MATRICES .

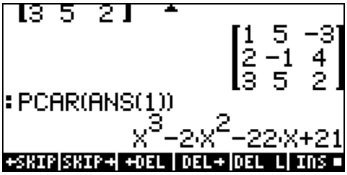




### Function PCAR

Function PCAR generates the characteristic polynomial of a square matrix using the contents of variable VX (a CAS reserved variable, typically equal to 'X') as the unknown in the polynomial. For example, enter the following matrix in ALG mode and find the characteristic equation using PCAR:

```
[[1,5,-3],[2,-1,4],[3,5,2]]
```

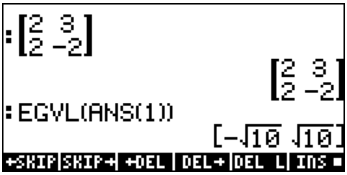


```
[3 5 2]
                    [1  5 -3]
                    [2 -1  4]
                    [3  5  2]
:PCAR(ANS(1))
        X^3-2·X^2-22·X+21
```

Using the variable λ to represent eigenvalues, this characteristic polynomial is to be interpreted as $\lambda^3$-2$\lambda^2$-22λ +21=0.

## Function EGVL

Function EGVL (EiGenVaLues) produces the eigenvalues of a square matrix. For example, the eigenvalues of the matrix shown below are calculated in ALG mode using function EGVL:

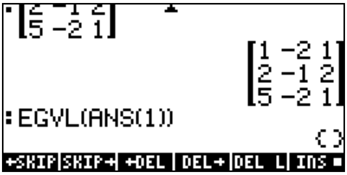


```
:[2  3]
 [2 -2]
                     [2  3]
                     [2 -2]
:EGVL(ANS(1))
              [-√10 √10]
```

The eigenvalues λ = [ -√10, √10 ].

**Note**: In some cases, you may not be able to find an 'exact' solution to the characteristic polynomial, and you will get an empty list as a result when using Function EGVL. If that were to happen to you, change the calculation mode to Approx in the CAS, and repeat the calculation.

For example, in exact mode, the following exercise produces an empty list as the solution:



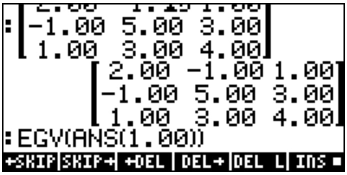
```
 [5 -2 1]
                    [1 -2 1]
                    [2 -1 2]
                    [5 -2 1]
:EGVL(ANS(1))
                          {}
```

Change mode to Approx and repeat the entry, to get the following eigenvalues: `[(1.38,2.22), (1.38,-2.22), (-1.76,0)]`.

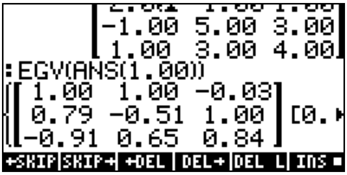
## Function EGV

Function EGV (EiGenValues and eigenvectors) produces the eigenvalues and eigenvectors of a square matrix. The eigenvectors are returned as the columns of a matrix, while the corresponding eigenvalues are the components of a vector.

For example, in ALG mode, the eigenvectors and eigenvalues of the matrix listed below are found by applying function EGV:

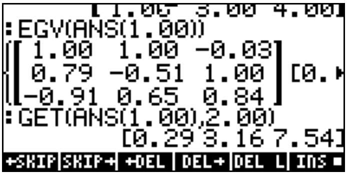


```
:│-1.00 5.00 3.00│
 [ 1.00  3.00 4.00]
       [ 2.00 -1.00 1.00]
       │-1.00 5.00 3.00│
       [ 1.00  3.00 4.00]
:EGV(ANS(1.00))
```



```
       │-1.00 5.00 3.00│
       [ 1.00  3.00 4.00]
:EGV(ANS(1.00))
{[[ 1.00  1.00 -0.03]
  [ 0.79 -0.51  1.00] [0.▸
  [-0.91  0.65  0.84]
```

The result shows the eigenvalues as the columns of the matrix in the result list. To see the eigenvalues we can use: GET(ANS(1),2), i.e., get the second element in the list in the previous result. The eigenvalues are:



```
:EGV(ANS(1.00))
{[[ 1.00  1.00 -0.03]
  [ 0.79 -0.51  1.00] [0.▸
  [-0.91  0.65  0.84]
:GET(ANS(1.00),2.00)
           [0.29 3.16 7.54]
```

In summary,

$$\lambda_1 = 0.29,\ \mathbf{x}_1 = [\ 1.00, 0.79, -0.91]^T,$$
$$\lambda_2 = 3.16,\ \mathbf{x}_2 = [1.00, -0.51,\ 0.65]^T,$$
$$\lambda_3 = 7.54,\ \mathbf{x}_1 = [-0.03,\ 1.00,\ 0.84]^T.$$

**Note**: A symmetric matrix produces all real eigenvalues, and its eigenvectors are mutually perpendicular. For the example just worked out, you can check that $\mathbf{x}_1 \bullet \mathbf{x}_2 = 0$, $\mathbf{x}_1 \bullet \mathbf{x}_3 = 0$, and $\mathbf{x}_2 \bullet \mathbf{x}_3 = 0$.

## Function JORDAN

Function JORDAN is intended to produce the diagonalization or Jordan-cycle decomposition of a matrix. In RPN mode, given a square matrix **A**, function JORDAN produces four outputs, namely:

- The minimum polynomial of matrix **A** (stack level 4)
- The characteristic polynomial of matrix **A** (stack level 3)
- A list with the eigenvectors corresponding to each eigenvalue of matrix **A** (stack level 2)
- A vector with the eigenvectors of matrix **A** (stack level 4)

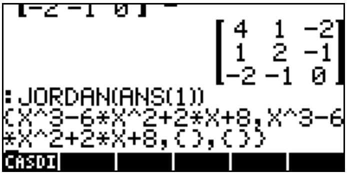
For example, try this exercise in RPN mode:

```
        [[4,1,-2],[1,2,-1],[-2,-1,0]]   JORDAN
```

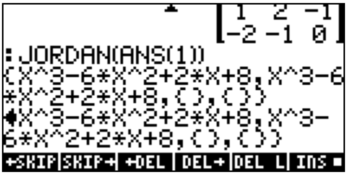
The output is the following:

```
4: 'X^3+-6*x^2+2*X+8'
3: 'X^3+-6*x^2+2*X+8'
2: { }
1: { }
```

The same exercise, in ALG mode, looks as in the following screen shots:



```
[-2 -1 0]
                    [ 4  1 -2]
                    [ 1  2 -1]
                    [-2 -1  0]
:JORDAN(ANS(1))
{X^3-6*X^2+2*X+8,X^3-6
*X^2+2*X+8,{},{}}
```



```
                    [ 1  2 -1]
                    [-2 -1  0]
:JORDAN(ANS(1))
{X^3-6*X^2+2*X+8,X^3-6
*X^2+2*X+8,{},{}}
◆X^3-6*X^2+2*X+8,X^3-
6*X^2+2*X+8,{},{}}
```

## Function MAD

This function, although not available in the EIGEN menu, also provides information related to the eigenvalues of a matrix. Function MAD is available through the MATRICES OPERATIONS sub-menu (← MATRICES ) and is intended to produce the adjoint matrix of a matrix.

In RPN mode, function MAD generate a number of properties of a square matrix, namely:

- the determinant (stack level 4)
- the formal inverse (stack level 3),
- in stack level 2, the matrix coefficients of the polynomial p($\mathbf{x}$) defined by $(\mathbf{x}\cdot\mathbf{I}\text{-}\mathbf{A})\ \cdot p(\mathbf{x})=m(\mathbf{x})\cdot\mathbf{I}$,
- the characteristic polynomial of the matrix (stack level 1)

Notice that the equation $(\mathbf{x}\cdot\mathbf{I}\text{-}\mathbf{A})\cdot p(\mathbf{x})=m(\mathbf{x})\cdot\mathbf{I}$ is similar, in form, to the eigenvalue equation $\mathbf{A}\cdot\mathbf{x} = \lambda\cdot\mathbf{x}$.

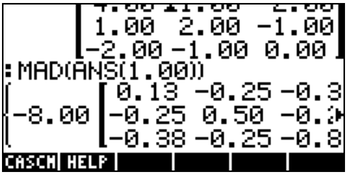
As an example, in RPN mode, try:

```
[[4,1,-2] [1,2,-1][-2,-1,0]] MAD
```

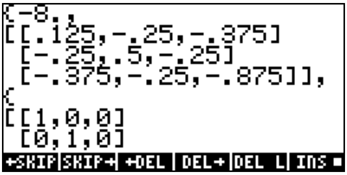
The result is:

```
4:  -8.
3:  [[ 0.13 –0.25 –0.38][-0.25 0.50 –0.25][-0.38 –0.25 –0.88]]
2: {[[1 0 0][0 1 0][0 0 1]] [[ -2 1 –2][1 –4 –1][-2 –1 –6] [[-1 2 3][2 –4 2][3 2 7]]}
1: ‘X^3+-6*x^2+2*X+8’
```

The same exercise, in ALG mode, will look as follows:



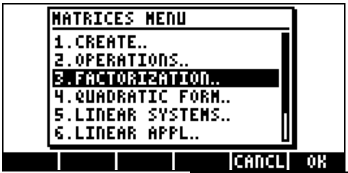
```
      1.00  2.00 -1.00
     -2.00 -1.00  0.00
:MAD(ANS(1.00))
        0.13 -0.25 -0.3
 -8.00 -0.25  0.50 -0.2
       -0.38 -0.25 -0.8
```

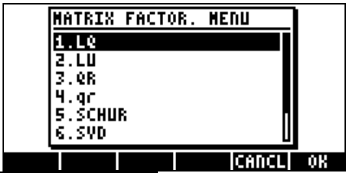


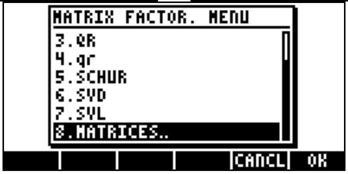
```
{-8.,
[[.125,-.25,-.375]
 [-.25,.5,-.25]
 [-.375,-.25,-.875]],
{
[[1,0,0]
 [0,1,0]
```

## Matrix factorization

Matrix factorization or decomposition consists of obtaining matrices that when multiplied result in a given matrix. We present matrix decomposition through the use of Functions contained in the matrix FACT menu. This menu is accessed through ↰ MATRICES .







Function contained in this menu are: LQ, LU, QR,SCHUR, SVD, SVL.

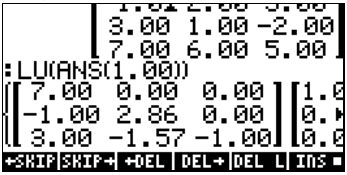
## Function LU

Function LU takes as input a square matrix **A**, and returns a lower-triangular matrix **L**, an upper triangular matrix **U**, and a permutation matrix **P**, in stack levels 3, 2, and 1, respectively. The results **L**, **U**, and **P**, satisfy the equation **P**·**A** = **L**·**U**. When you call the LU function, the calculator performs a Crout LU decomposition of **A** using partial pivoting.

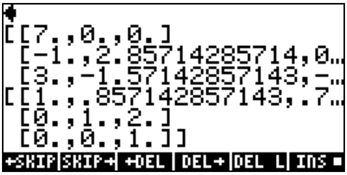
For example, in RPN mode: `[[-1,2,5][3,1,-2][7,6,5]] LU` produces:

```
3:[[7 0 0][-1 2.86 0][3 –1.57 –1]
2: [[1 0.86 0.71][0 1 2][0 0 1]]
1: [[0 0 1][1 0 0][0 1 0]]
```

In ALG mode, the same exercise will be shown as follows:



```
 3.00  1.00 -2.00
 7.00  6.00  5.00
:LU(ANS(1.00))
{[ 7.00  0.00  0.00 ][1.0
  -1.00  2.86  0.00 ][0.▸
   3.00 -1.57 -1.00 ][0.0
```



```
[[7.,0.,0.]
 [-1.,2.85714285714,0…
 [3.,-1.57142857143,-…
[[1.,.857142857143,.7…
 [0.,1.,2.]
 [0.,0.,1.]]
```

## Orthogonal matrices and singular value decomposition

A square matrix is said to be orthogonal if its columns represent unit vectors that are mutually orthogonal. Thus, if we let matrix **U** = [$\mathbf{v}_1$ $\mathbf{v}_2$ … $\mathbf{v}_n$] where the $\mathbf{v}_i$, i = 1, 2, …, n, are column vectors, and if $\mathbf{v}_i \bullet \mathbf{v}_j = \delta_{ij}$, where $\delta_{ij}$ is the Kronecker's delta function, then **U** will be an orthogonal matrix. This conditions also imply that $\mathbf{U}\cdot \mathbf{U}^T = \mathbf{I}$.

The Singular Value Decomposition (SVD) of a rectangular matrix $\mathbf{A}_{m\times n}$ consists in determining the matrices **U**, **S**, and **V**, such that $\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m} \cdot \mathbf{S}_{m\times n} \cdot \mathbf{V}^{T}_{n\times n}$, where **U** and **V** are orthogonal matrices, and **S** is a diagonal matrix. The diagonal elements of **S** are called the <u>*singular values*</u> of **A** and are usually ordered so that $s_i \geq s_{i+1}$, for *i = 1, 2, …, n-1*. The columns [$\mathbf{u}_j$] of **U** and [$\mathbf{v}_j$] of **V** are the corresponding <u>*singular vectors*</u>.

**Function SVD**

In RPN, function SVD (Singular Value Decomposition) takes as input a matrix $\mathbf{A}_{n\times m}$, and returns the matrices $\mathbf{U}_{n\times n}$, $\mathbf{V}_{m\times m}$, and a vector **s** in stack levels 3, 2, and 1, respectively. The dimension of vector **s** is equal to the minimum of the values n and m. The matrices **U** and **V** are as defined earlier for singular value decomposition, while the vector **s** represents the main diagonal of the matrix **S** used earlier.

For example, in RPN mode: `[[5,4,-1],[2,-3,5],[7,2,8]] SVD`

```
3: [[-0.27 0.81 –0.53][-0.37 –0.59 –0.72][-0.89 3.09E-3 0.46]]
2: [[ -0.68 –0.14 –0.72][ 0.42 0.73 –0.54][-0.60 0.67 0.44]]
1: [ 12.15 6.88 1.42]
```

**Function SVL**

Function SVL (Singular VaLues) returns the singular values of a matrix $\mathbf{A}_{n\times m}$ as a vector **s** whose dimension is equal to the minimum of the values n and m. For example, in RPN mode, `[[5,4,-1],[2,-3,5],[7,2,8]] SVL` produces [ 12.15 6.88 1.42].

## Function SCHUR

In RPN mode, function SCHUR produces the *Schur decomposition* of a square matrix **A** returning matrices **Q** and **T**, in stack levels 2 and 1, respectively, such that $\mathbf{A} = \mathbf{Q}\cdot\mathbf{T}\cdot\mathbf{Q}^{T}$, where **Q** is an orthogonal matrix, and **T** is a triangular matrix. For example, in RPN mode,

```
[[2,3,-1][5,4,-2][7,5,4]] SCHUR
```

results in:

```
2: [[0.66 –0.29 –0.70][-0.73 –0.01 –0.68][ -0.19 –0.96 0.21]]
```

```
1: [[-1.03 1.02 3.86 ][ 0 5.52 8.23 ][ 0 –1.82 5.52]]
```

## Function LQ

The LQ function produces the *LQ factorization* of a matrix $\mathbf{A}_{n\times m}$ returning a lower $\mathbf{L}_{n\times m}$ trapezoidal matrix, a $\mathbf{Q}_{m\times m}$ orthogonal matrix, and a $\mathbf{P}_{n\times n}$ permutation matrix, in stack levels 3, 2, and 1. The matrices **A**, **L**, **Q** and **P** are related by $\mathbf{P}\cdot\mathbf{A} = \mathbf{L}\cdot\mathbf{Q}$. (A trapezoidal matrix out of an n×m matrix is the equivalent of a triangular matrix out of an n×n matrix). For example,

```
        [[ 1, -2, 1][ 2, 1, -2][ 5, -2, 1]] LQ
```

produces

```
3: [[-5.48 0 0][-1.10 –2.79 0][-1.83 1.43 0.78]]
2: [[-0.91 0.37 -0.18] [-0.36 -0.50 0.79] [-0.20 -0.78 -0.59]]
1: [[0 0 1][0 1 0][1 0 0]]
```

## Function QR

In RPN, function QR produces the *QR factorization* of a matrix $\mathbf{A}_{n\times m}$ returning a $\mathbf{Q}_{n\times n}$ orthogonal matrix, a $\mathbf{R}_{n\times m}$ upper trapezoidal matrix, and a $\mathbf{P}_{m\times m}$ permutation matrix, in stack levels 3, 2, and 1. The matrices **A**, **P**, **Q** and **R** are related by $\mathbf{A}\cdot\mathbf{P} = \mathbf{Q}\cdot\mathbf{R}$. For example,

```
[[ 1,-2,1][ 2,1,-2][ 5,-2,1]]  QR
```

produces

```
3: [[-0.18 0.39 0.90][-0.37 –0.88 0.30][-0.91 0.28 –0.30]]
2: [[ -5.48 –0.37 1.83][ 0 2.42 –2.20][0 0 –0.90]]
1: [[1 0 0][0 0 1][0 1 0]]
```

**Note:** Examples and definitions for all functions in this menu are available through the help facility in the calculator. Try these exercises in ALG mode to see the results in that mode.

# Matrix Quadratic Forms

A <u>quadratic form</u> from a square matrix **A** is a polynomial expression originated from $\mathbf{x}\cdot\mathbf{A}\cdot\mathbf{x}^T$. For example, if we use **A** = [[2,1,–1][5,4,2][3,5,–1]], and **x** = [X Y Z]$^T$, the corresponding quadratic form is calculated as

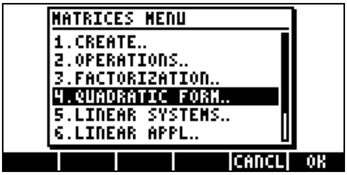
$$\mathbf{x}\cdot\mathbf{A}\cdot\mathbf{x}^T = \begin{bmatrix} X & Y & Z \end{bmatrix}\cdot\begin{bmatrix} 2 & 1 & -1 \\ 5 & 4 & 2 \\ 3 & 5 & -1 \end{bmatrix}\cdot\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

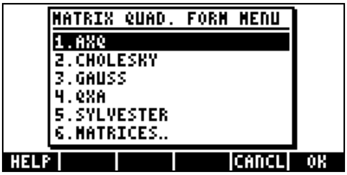
$$= \begin{bmatrix} X & Y & Z \end{bmatrix}\cdot\begin{bmatrix} 2X+Y-Z \\ 5X+4Y+2Z \\ 3X+5Y-Z \end{bmatrix}$$

Finally, $\mathbf{x}\cdot\mathbf{A}\cdot\mathbf{x}^T = 2X^2+4Y^2-Z^2+6XY+2XZ+7ZY$

## The QUADF menu

The HP 49 G calculator provides the QUADF menu for operations related to QUADratic Forms. The QUADF menu is accessed through ← MATRICES .





This menu includes functions AXQ, CHOLESKY, GAUSS, QXA, and SYLVESTER.

### Function AXQ

In RPN mode, function AXQ produces the quadratic form corresponding to a matrix $\mathbf{A}_{n\times n}$ in stack level 2 using the *n* variables in a vector placed in stack level 1. Function returns the quadratic form in stack level 1 and the vector of variables in stack level 1. For example,

```
[[2,1,-1],[5,4,2],[3,5,-1]] ENTER
['X','Y','Z'] ENTER AXQ
```

returns

2: '2*X^2+(6*Y+2*Z)*X+4*Y^2+7*Z*y-Z^2'
1: ['X' 'Y' 'Z']

### Function QXA

Function QXA takes as arguments a quadratic form in stack level 2 and a vector of variables in stack level 1, returning the square matrix **A** from which the quadratic form is derived in stack level 2, and the list of variables in stack level 1. For example,

```
'X^2+Y^2-Z^2+4*X*Y-16*X*Z' ENTER
['X','Y','Z'] ENTER QXA
```

returns

2: [[1 2 –8][2 1 0][-8 0 –1]]

1: ['X' 'Y' 'Z']

### Diagonal representation of a quadratic form

Given a symmetric square matrix **A**, it is possible to "diagonalize" the matrix **A** by finding an orthogonal matrix **P** such that $\mathbf{P}^T\cdot\mathbf{A}\cdot\mathbf{P} = \mathbf{D}$, where **D** is a diagonal matrix. If $Q = \mathbf{x}\cdot\mathbf{A}\cdot\mathbf{x}^T$ is a quadratic form based on **A**, it is possible to write the quadratic form Q so that it only contains square terms from a variable **y**, such that $\mathbf{x} = \mathbf{P}\cdot\mathbf{y}$, by using $Q = \mathbf{x}\cdot\mathbf{A}\cdot\mathbf{x}^T = (\mathbf{P}\cdot\mathbf{y})\cdot\mathbf{A}\cdot (\mathbf{P}\cdot\mathbf{y})^T = \mathbf{y}\cdot(\mathbf{P}^T\cdot\mathbf{A}\cdot\mathbf{P})\cdot\mathbf{y}^T = \mathbf{y}\cdot\mathbf{D}\cdot\mathbf{y}^T$.

### Function SYLVESTER

Function SYLVESTER takes as argument a symmetric square matrix **A** and returns a vector containing the diagonal terms of a diagonal matrix **D**, and a matrix **P**, so that $\mathbf{P}^T\cdot\mathbf{A}\cdot\mathbf{P} = \mathbf{D}$. For example,

```
[[2,1,-1],[1,4,2],[-1,2,-1]] SYLVESTER
```

produces

2: [ 1/2 2/7 -23/7]

1: [[2 1 –1][0 7/2 5/2][0 0 1]]

### Function GAUSS

Function GAUSS returns the diagonal representation of a quadratic form $Q = \mathbf{x}\cdot\mathbf{A}\cdot\mathbf{x}^T$ taking as arguments the quadratic form in stack level 2 and the vector of variables in stack level 1. The result of this function call is the following:

- An array of coefficients representing the diagonal terms of **D** (stack level 4)
- A matrix **P** such that $\mathbf{A} = \mathbf{P}^T\cdot\mathbf{D}\cdot\mathbf{P}$ (stack level 3)
- The diagonalized quadratic form (stack level 2)

- The list of variables (stack level 1)

For example,

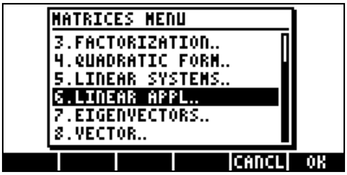
```
'X^2+Y^2-Z^2+4*X*Y-16*X*Z' ENTER
    ['X','Y','Z'] ENTER GAUSS
```

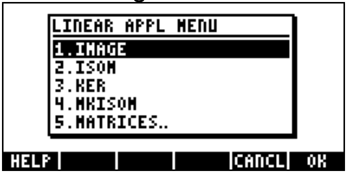
returns

4: [1 –0.333 20.333]
3: [[1 2 –8][0 –3 16][0 0 1]]
2: '61/3*Z^2+ -1/3*(16*Z+-3*Y)^2+(-8*z+2*Y+X)^2'
1: ['X' 'Y' 'Z']

## Linear Applications

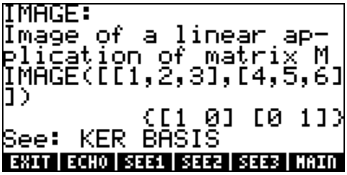
The LINEAR APPLICATIONS menu is available through the ← MATRICES .



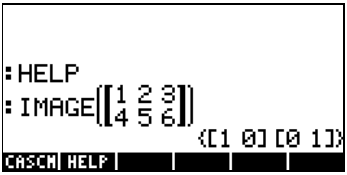


Information on the functions listed in this menu is presented below by using the calculator's own help facility. The figures show the help facility entry and the attached examples.

### Function IMAGE

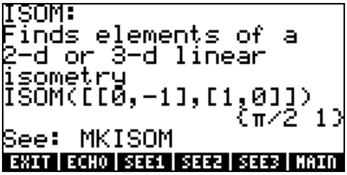


```
IMAGE:
Image of a linear ap-
plication of matrix M
IMAGE([[1,2,3],[4,5,6]
])
          {[1 0] [0 1]}
See: KER BASIS
```

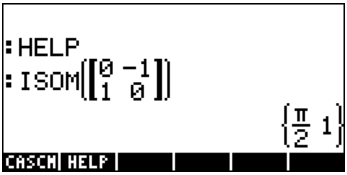


```
:HELP
:IMAGE([[1 2 3][4 5 6]])
                {[1 0] [0 1]}
```

### Function ISOM

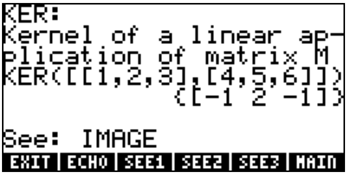


```
ISOM:
Finds elements of a
2-d or 3-d linear
isometry
ISOM([[0,-1],[1,0]])
               {π/2 1}
See: MKISOM
```

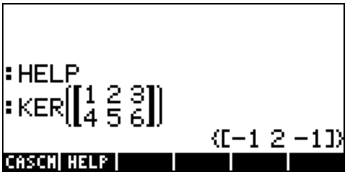


```
:HELP
:ISOM([[0 -1][1 0]])
                 {π/2 1}
```

## Function KER

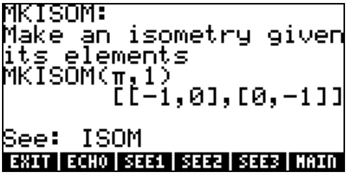


```
KER:
Kernel of a linear ap-
plication of matrix M
KER([[1,2,3],[4,5,6]])
              {[-1 2 -1]}

See: IMAGE
```

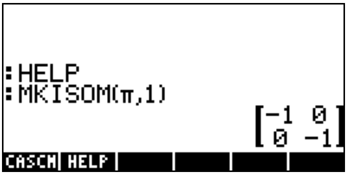


```
:HELP
:KER([1 2 3]
     [4 5 6])
                {[-1 2 -1]}
```

## Function MKISOM



```
MKISOM:
Make an isometry given
its elements
MKISOM(π,1)
         [[-1,0],[0,-1]]

See: ISOM
```



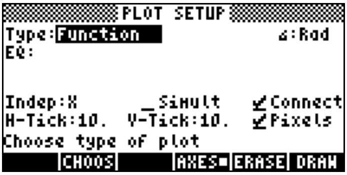
```
:HELP
:MKISOM(π,1)
                    [-1  0]
                    [ 0 -1]
```

# Chapter 12
# Graphics

In this chapter we introduce some of the graphics capabilities of the calculator. We will present graphics of functions in Cartesian coordinates and polar coordinates, parametric plots, graphics of conics, bar plots, scatterplots, and a variety of three-dimensional graphs.

## Graphs options in the calculator

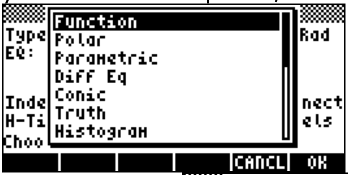
To access the list of graphic formats available in the calculator, use the keystroke sequence ← 2D/3D (F4) Please notice that if you are using the RPN mode these two keys must be pressed simultaneously to activate any of the graph functions. After activating the 2D/3D function, the calculator will produce the PLOT SETUP window, which includes the TYPE field as illustrated below.

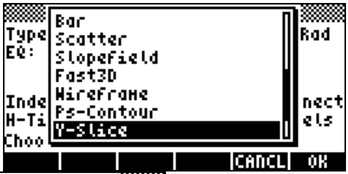


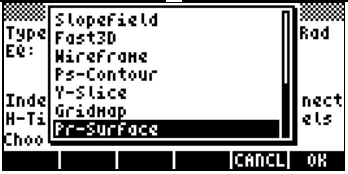
PLOT SETUP

Type: Function ∡:Rad
EQ:
Indep:X _Simult ✓Connect
H-Tick:10. V-Tick:10. ✓Pixels
Choose type of plot

Right in front of the TYPE field you will, most likely, see the option *Function* highlighted. This is the default type of graph for the calculator. To see the list of available graph types, press the soft menu key labeled CHOOS. This will produce a drop down menu with the following options (use the up- and down-arrow keys to see all the options):







These graph options are described briefly next.

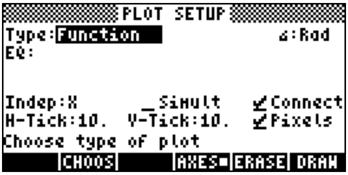
*Function*: for equations of the form y = f(x) in plane Cartesian coordinates
*Polar*: for equations of the from r = f(θ) in polar coordinates in the plane
*Parametric*: for plotting equations of the form x = x(t), y = y(t) in the plane
*Diff Eq*: for plotting the numerical solution of a linear differential equation
*Conic*: for plotting conic equations (circles, ellipses, hyperbolas, parabolas)
*Truth*: for plotting inequalities in the plane
*Histogram*: for plotting frequency histograms (statistical applications)
*Bar*: for plotting simple bar charts
*Scatter*: for plotting scatter plots of discrete data sets (statistical applications)
*Slopefield*: for plotting traces of the slopes of a function f(x,y) = 0.
*Fast3D*: for plotting curved surfaces in space
*Wireframe*: for plotting curved surfaces in space showing wireframe grids
*Ps-Contour*: for plotting contour plots of surfaces
*Y- Slice*: for plotting a slicing view of a function f(x,y).
*Gridmap*: for plotting real and imaginary part traces of a complex function
*Pr-Surface*: for parametric surfaces given by x = x(u,v), y = y(u,v), z = z(u,v).

## Plotting an expression of the form *y = f(x)*

In this section we present an example of a plot of a function of the form *y = f(x).* In order to proceed with the plot, first, purge the variable x, if it is defined in the current directory (x will be the independent variable in the calculator's PLOT feature, therefore, you don't want to have it pre-defined). Create a sub-directory called 'TPLOT' (for test plot), or other meaningful name, to perform the following exercise.
As an example, let's plot the function,

$$f(x)=\frac{1}{\sqrt{2\pi}}\exp(-\frac{x^2}{2})$$

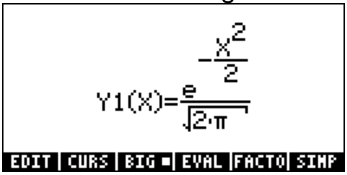
- First, enter the PLOT SETUP environment by pressing, ↰ 2D/3D . Make sure that the option Function is selected as the TYPE, and that ‘X’ is selected as the independent variable (INDEP). Press NXT OK to return to normal calculator display. The PLOT SET UP window should look similar to this:



```
PLOT SETUP
Type:Function                ∡:Rad
EQ:

Indep:X       _Simult     ✓Connect
H-Tick:10.  V-Tick:10.    ✓Pixels
Choose type of plot
```

- **Note**: You will notice that a new variable, called PPAR, shows up in your soft menu key labels. This stands for Plot PARameters. To see its contents, press ↱ PPAR. A detailed explanation of the contents of PPAR is provided later in this Chapter. Press ◀ to drop this line from the stack.

- Enter the PLOT environment by pressing ↰ Y= (press them simultaneously if in RPN mode). Press ADD to get you into the equation writer. You will be prompted to fill the right-hand side of an equation Y1(x) = ▪. Type the function to be plotted so that the Equation Writer shows the following:



$$Y1(X)=\frac{e^{-\frac{X^2}{2}}}{\sqrt{2\cdot\pi}}$$

- Press ENTER to return to the PLOT SETUP window. The expression 'Y1(X) = EXP(-X^2/2)/√(2*π)' will be highlighted. Press NXT OK to return to normal calculator display.

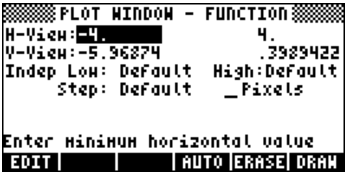
**Note**: Two new variables show up in your soft menu key labels, namely EQ and Y1. To see the contents of EQ, use ↱ EQ. The content of EQ is simply the function name 'Y1(X)'. The variable EQ is used by the calculator to store the equation, or equations, to plot.

To see the contents of Y1 press ↱ Y1. You will get the function Y1(X) defined as the program:

```
<< →X 'EXP(-X^2/2)/√(2*π)' >>.
```

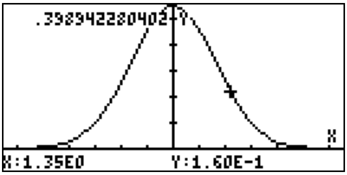
Press ◄, twice, to drop the contents of the stack.

- Enter the PLOT WINDOW environment by entering ↰ WIN (press them simultaneously if in RPN mode). Use a range of –4 to 4 for H-VIEW, then press AUTO to generate the V-VIEW automatically. The PLOT WINDOW screen looks as follows:



```
PLOT WINDOW - FUNCTION
H-View:-4.              4.
V-View:-5.96874         .3989422
Indep Low: Default   High:Default
     Step: Default   _Pixels

Enter minimum horizontal value
EDIT |    |    |AUTO |ERASE|DRAW
```

- Plot the graph: ERASE DRAW (wait till the calculator finishes the graphs)
- To see labels: EDIT NXT LABEL MENU
- To recover the first graphics menu: NXT NXT PICT
- To trace the curve: TRACE (X,Y) . Then use the right- and left-arrow keys (◄►) to move about the curve. The coordinates of the points you trace will be shown at the bottom of the screen. Check that for x = 1.05 , y = 0.231. Also, check that for x = -1.48 , y = 0.134. Here is picture of the graph in tracing mode:



- To recover the menu, and return to the PLOT WINDOW environment, press NXT CANCL, then NXT OK.

## Some useful PLOT operations for FUNCTION plots

In order to discuss these PLOT options, we'll modify the function to force it to have some real roots (Since the current curve is totally contained above the x axis, it has no real roots.)  Press ↱ Y1 to list the contents of the function Y1 on the stack: << →X 'EXP(-X^2/2)/ √(2*π) ' >>.  To edit this expression use:

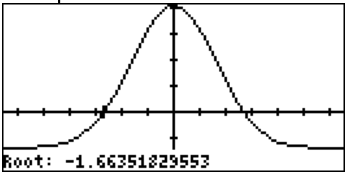
| | |
|---|---|
| ▼ | Launches the line editor |
| ↱ ▼ | Moves cursor to the end of the line |
| ◀ ◀ ◀ – 0 • 1 | Modifies the expression |
| ENTER | Returns to calculator display |

Next, store the modified expression into variable y by using ↰ Y1 if in RPN mode, or ↰ ANS STO▶ Y1 in ALG mode.

The function to be plotted is now, $f(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}) - 0.1$

Enter the PLOT WINDOW environment by entering ↰ WIN (press them simultaneously if in RPN mode.)  Keep the range of –4 to 4 for H-VIEW, press ▼ AUTO to generate the V-VIEW. To plot the graph, press ERASE DRAW

- Once the graph is plotted, press FCN to access the *function* menu. With this menu you can obtain additional information about the plot such as intersects with the x-axis, roots, slopes of the tangent line, area under the curve, etc.
- For example, to find the root on the left side of the curve, move the cursor near that point, and press ROOT.  You will get the result:  ROOT: -1.6635....  Press NXT to recover the menu.  Here is the result of ROOT in the current plot:



Root: -1.66351829553

- If you move the cursor towards the right-hand side of the curve, by pressing the right-arrow key (▶), and press ROOT, the result now is ROOT: 1.6635... The calculator indicated, before showing the root, that it was found through *SIGN REVERSAL.* Press NXT to recover the menu.
- Pressing ISECT will give you the intersection of the curve with the x-axis, which is essentially the root. Place the cursor exactly at the root and press ISECT. You will get the same message as before, namely *SIGN REVERSAL*, before getting the result I-SECT: 1.6635…. The ISECT function is intended to determine the intersection of any two curves closest to the location of the cursor. In this case, where only one curve, namely, Y1(X), is involved, the intersection sought is that of f(x) with the x-axis, however, you must place the cursor right at the root to produce the same result. Press NXT to recover the menu.
- Place the cursor on the curve at any point and press SLOPE to get the value of the slope at that point. For example, at the negative root, SLOPE: 0.16670…. Press NXT to recover the menu.
- To determine the highest point in the curve, place the cursor near the vertex and press EXTR The result is EXTRM: 0.. Press NXT to recover the menu.
- Other buttons available in the first menu are AREA to calculate the area under the curve, and SHADE to shade an area under the curve. Press NXT to see more options. The second menu includes one button called VIEW that flashes for a few seconds the equation plotted. Press VIEW. Alternatively, you can press the button NEXQ (NEXt eQuation) to see the name of the function Y1(x). Press NXT to recover the menu.
- The button F(X) gives the value of f(x) corresponding to the cursor position. Place the cursor anywhere in the curve and press F(X). The value will be shown in the lower left corner of the display. Press NXT to recover the menu.
- Place the cursor in any given point of the trajectory and press TANL to obtain the equation of the tangent line to the curve at that point. The equation will be displayed on the lower left corner of the display. Press NXT to recover the menu.
- If you press F' the calculator will plot the derivative function, f'(x) = df/dx, as well as the original function, f(x). Notice that the two

curves intercept at two points. Move the cursor near the left intercept point and press FCN ISECT, to get I-SECT: (-0.6834…,0.21585). Press NXT to recover the menu.

- To leave the FCN environment, press PICT (or NXT PICT).
- Press CANCL to return to the PLOT WINDOW environment. Then, press NXT OK to return to normal calculator display.

**Note**: the stack will show all the graph operations performed, properly identified.

- Enter the PLOT environment by pressing, simultaneously if in RPN mode, ← Y=. Notice that the highlighted field in the PLOT environment now contains the derivative of Y1(X). Press NXT OK to return to return to normal calculator display.
- Press → EQ to check the contents of EQ. You will notice that it contains a list instead of a single expression. The list has as elements an expression for the derivative of Y1(X) and Y1(X) itself. Originally, EQ contained only Y1(x). After we pressed F' in the FCN environment, the calculator automatically added the derivative of Y1(x) to the list of equations in EQ.

## Saving a graph for future use

If you want to save your graph to a variable, get into the PICTURE environment by pressing ◄. Then, press EDIT NXT NXT PICT→. This captures the current picture into a graphics object. To return to the stack, press PICT CANCL.

In level 1 of the stack you will see a graphics object described as `Graphic 131 × 64`. This can be stored into a variable name, say, PIC1.

To display your figure again, recall the contents of variable PIC1 to the stack. The stack will show the line: `Graphic 131 × 64`. To see the graph, enter the PICTURE environment, by pressing ◄.

Clear the current picture, EDIT NXT ERASE.

Move the cursor to the upper left corner of the display, by using the ◄ and ▲ keys.

To display the figure currently in level 1 of the stack press NXT REPL .

To return to normal calculator function, press PICT CANCL.

**Note:** To save printing space, we will not include more graphs produced by following the instructions in this Chapter. The user is invited to produce those graphics on his or her own.

## Graphics of transcendental functions

In this section we use some of the graphics features of the calculator to show the typical behavior of the natural log, exponential, trigonometric and hyperbolic functions. You will not see more graphs in this chapter, instead the user should see them in the calculator.

### Graph of ln(X)

Press, simultaneously if in RPN mode, the left-shift key ↰ and the 2D/3D (F4) key to produce the PLOT SETUP window. The field labeled **`Type`** will be highlighted. If the option `Function` is not already selected press the soft key labeled CHOOS, use the up and down keys to select `Function`, and press OK to complete the selection. Check that the field labeled Indep: contains the variable 'X'. If that is not so, press the down arrow key twice until the `Indep` field is highlighted, press the soft key labeled EDIT and modify the value of the independent variable to read 'X'. Press OK when done. Press NXT OK to return to normal calculator display.

Next, we'll <u>resize the plot window</u>. First, press, simultaneously if in RPN mode, the left-shift key ↰ and the Y= (F1) key to produce the PLOT-FUNCTION window. If there is any equation highlighted in this window, press DEL as needed to clear the window completely. When the PLOT-FUNCTION window is empty you will get a prompt message that reads: **`No Equ., Press ADD`**. Press the soft key labeled ADD . This will trigger the equation writer with the expression Y1(X)=◀ . Type LN(X). Press ENTER to return to the PLOT-FUNCTION window. Press NXT OK to return to normal calculator display.

The next step is to press, simultaneously if in RPN mode, the left-shift key ↰ and the WIN (F2) key to produce the PLOT WINDOW - FUNCTION window. Most likely, the display will show the horizontal (H-View) and vertical (`V-View`) ranges as: H-View: -6.5 6.5, V-View: -3.1 3.2

These are the default values for the x- and y-range, respectively, of the current graphics display window. Next, change the H-View values to read: H-View: -1 10 by using 1 +/- OK 1 0 OK. Next, press the soft key labeled AUTO to let the calculator determine the corresponding vertical range. After a couple of seconds this range will be shown in the PLOT WINDOW-FUNCTION window. At this point we are ready to produce the graph of ln(X). Press ERASE DRAW to plot the natural logarithm function.

To add labels to the graph press EDIT NXT LABEL. Press MENU to remove the menu labels, and get a full view of the graph. Press NXT to recover the current graphic menu. Press NXT PICT to recover the original graphical menu.

To determine the coordinates of points on the curve press TRACE (the cursor moves on top of the curve at a point located near the center of the horizontal range). Next, press (X,Y) to see the coordinates of the current cursor location. These coordinates will be shown at the bottom of the screen. Use the right- and left-arrow keys to move the cursor along the curve. As you move the cursor along the curve the coordinates of the curve are displayed at the bottom of the screen. Check that when Y:1.00E0, X:2.72E0. This is the point *(e,1)*, since *ln(e) = 1*. Press NXT to recover the graphics menu.

Next, we will find the intersection of the curve with the x-axis by pressing FCN ROOT. The calculator returns the value `Root: 1`, confirming that *ln(1) = 0*. Press NXT NXT PICT CANCL to return to the PLOT WINDOW – FUNCTION. Press ENTER to return to normal calculator display. You will notice that the root found in the graphics environment was copied to the calculator stack.

**Note**: When you press VAR , your variables list will show new variables called X and Y1 . Press ↱Y1 to see the contents of this variable. You will get the program << → X 'LN(X)' >> , which you will recognize as the program that may result from defining the function 'Y1(X) = LN(X)' by using ↰DEF . This is basically what happens when you ADD a function in the PLOT – FUNCTION window (the window that results from pressing ↰ Y= , simultaneously if in RPN mode), i.e., the function gets defined and added to your variable list.

Next, press ↱X to see the contents of this variable. A value of 10.275 is placed in the stack. This value is determined by our selection for the horizontal display range. We selected a range between -1 and 10 for X. To produce the graph, the calculator generates values between the range limits using a constant increment, and storing the values generated, one at a time, in the variable X as the graph is drawn. For the horizontal range ( –1,10), the increment used seems to be 0.275. When the value of X becomes larger than the maximum value in the range (in this case, when X = 10.275), the drawing of the graph stops. The last value of X for the graphic under consideration is kept in variable X. Delete X and Y1 before continuing.

## Graph of the exponential function

First, load the function *exp(X),* by pressing, simultaneously if in RPN mode, the left-shift key ↰ and the Y= (EEX) key to access the PLOT-FUNCTION window. Press DEL to remove the function LN(X), if you didn't delete Y1 as suggested in the previous note. Press ADD and type ↰$e^x$ ALPHA X ENTER to enter EXP(X) and return to the PLOT-FUNCTION window. Press NXT OK to return to normal calculator display.

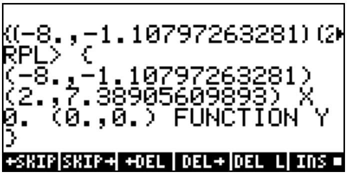
Next, press, simultaneously if in RPN mode, the left-shift key ↰ and the WIN (F2) key to produce the PLOT WINDOW - FUNCTION window. Change the H-View values to read: H-View: -8 2

by using 8 +/- OK 2 OK. Next, press AUTO. After the vertical range is calculated, press ERASE DRAW to plot the exponential function.

To add labels to the graph press EDIT NXT LABEL. Press MENU to remove the menu labels, and get a full view of the graph. Press NXT NXT PICT CANCL to return to the PLOT WINDOW – FUNCTION. Press ENTER to return to normal calculator display.

## The PPAR variable

Press VAR to recover your variables menu, if needed. In your variables menu you should have a variable labeled PPAR . Press ↱ PPAR to get the contents of this variable in the stack. Press the down-arrow key, , to launch the stack editor, and use the up- and down-arrow keys to view the full contents of PPAR. The screen will show the following values:



```
{(-8.,-1.10797263281)(2▸
RPL> {
(-8.,-1.10797263281)
(2.,7.38905609893) X
0. (0.,0.) FUNCTION Y
}
```

PPAR stands for *Plot PARameters*, and its contents include two ordered pairs of real numbers, (-8.,-1.10797263281) and (2.,7.38905609893),

which represent the *coordinates of the lower left corner and the upper right corner* of the plot, respectively. Next, PPAR lists the *name of the independent variable*, X, followed by a number that specifies the *increment of the independent variable* in the generation of the plot. The value shown here is the default value, zero (0.), which specifies increments in X corresponding to 1 pixel in the graphics display. The next element in PPAR is a *list containing first the coordinates of the point of intersection of the plot axes*, i.e., (0.,0.), *followed by a list that specifies the tick mark annotation* on the x- and y-axes, respectively {# 10d # 10d}. Next, PPAR lists the *type of plot* that is to be generated, i.e., FUNCTION, and, finally, *the y-axis label*, i.e., Y.

The variable PPAR, if non-existent, is generated every time you create a plot. The contents of the function will change depending on the type of plot and on the options that you select in the PLOT window (the window generated by the simultaneous activation of the ↰ and WIN (F2) keys.

## Inverse functions and their graphs

Let *y = f(x)*, if we can find a function *y = g(x)*, such that, *g(f(x)) = x*, then we say that *g(x)* is the *inverse function* of *f(x)*. Typically, the notation $g(x) = f^{-1}(x)$ is used to denote an inverse function. Using this notation we can write: if *y = f(x)*, then $x = f^{-1}(y)$. Also, $f(f^{-1}(x)) = x$, and $f^{-1}(f(x)) = x$.

As indicated earlier, the ln(x) and exp(x) functions are inverse of each other, i.e., ln(exp(x)) = x, and exp(ln(x)) = x. This can be verified in the calculator by typing and evaluating the following expressions in the Equation Writer: LN(EXP(X)) and EXP(LN(X)). They should both evaluate to X.

When a function *f(x)* and its inverse $f^{-1}(x)$ are plotted simultaneously in the same set of axes, their graphs are reflections of each other about the line *y = x*. Let's check this fact with the calculator for the functions LN(X) and EXP(X) by following this procedure:

Press, simultaneously if in RPN mode, ← Y= . The function Y1(X) = EXP(X) should be available in the PLOT - FUNCTION window from the previous exercise. Press ADD , and type the function Y2(X) = LN(X). Also, load the function Y3(X) = X. Press NXT OK to return to normal calculator display.

Press, simultaneously if in RPN mode, ← WIN , and change the H-View range to read: H-View: -8 8

Press AUTO to generate the vertical range. Press ERASE DRAW to produce the graph of y = ln(x), y = exp(x), and y =x, simultaneously if in RPN mode.

You will notice that only the graph of y = exp(x) is clearly visible. Something went wrong with the AUTO selection of the vertical range. What happens is that, when you press AUTO in the PLOT FUNCTION – WINDOW screen, the calculator produces the vertical range corresponding to the first function in the list of functions to be plotted. Which, in this case, happens to be Y1(X) = EXP(X). We will have to enter the vertical range ourselves in order to display the other two functions in the same plot.

Press CANCL to return to the PLOT FUNCTION – WINDOW screen. Modify the vertical and horizontal ranges to read: H-View: -8 8, V-View: -4 4

By selecting these ranges we ensure that the scale of the graph is kept 1 vertical to 1 horizontal. Press ERASE DRAW and you will get the plots of the natural logarithm, exponential, and *y* = *x* functions. It will be evident from the graph that LN(X) and EXP(X) are reflections of each other about the line y = X. Press CANCL to return to the PLOT WINDOW – FUNCTION. Press ENTER to return to normal calculator display.

## Summary of FUNCTION plot operation

In this section we present information regarding the PLOT SETUP, PLOT-FUNCTION, and PLOT WINDOW screens accessible through the left-shift key combined with the soft-menu keys F1 through F4 . Based on the graphing examples presented above, the procedure to follow to produce a FUNCTION plot (i.e., one that plots one or more functions of the form Y = F(X)), is the following:

↰ 2D/3D , simultaneously if in RPN mode: Access to the PLOT SETUP window. If needed, change `TYPE` to `FUNCTION`, and enter the name of the independent variable.

*Settings*:

- A check on `_Simult` means that if you have two or more plots in the same graph, they will be plotted simultaneously when producing the graph.
- A check on `_Connect` means that the curve will be a continuous curve rather than a set of individual points.
- A check on `_Pixels` means that the marks indicated by `H-Tick` and V-Tick will be separated by that many pixels.
- The default value for both by `H-Tick` and `V-Tick` is 10.

*Soft key menu options*:

- Use EDIT to edit functions of values in the selected field.
- Use CHOOS to select the type of plot to use when the `Type:` field is highlighted. For the current exercises, we want this field set to FUNCTION.

**Note**: the soft menu keys EDIT and CHOOS are not available at the same time. One or the other will be selected depending on which input field is highlighted.

- Press the AXES soft menu key to select or deselect the plotting of axes in the graph. If the option 'plot axes' is selected, a square dot will appear in the key label: AXES■ . Absence of the square dot indicates that axes will not be plotted in the graph.
- Use ERASE to erase any graph currently existing in the graphics display window.
- Use DRAW to produce the graph according to the current contents of PPAR for the equations listed in the PLOT-FUNCTION window.
- Press NXT to access the second set of soft menu keys in this screen.
- Use RESET to reset any selected field to its default value.
- Use CANCL to cancel any changes to the PLOT SETUP window and return to normal calculator display.
- Press OK to save changes to the options in the PLOT SETUP window and return to normal calculator display.

↰ Y=, simultaneously if in RPN mode: . Access to the PLOT window (in this case it will be called PLOT –FUNCTION window).

*Soft menu key options*:

- Use EDIT to edit the highlighted equation.
- Use ADD to add new equations to the plot.

**Note**: ADD or EDIT will trigger the equation writer EQW that you can use to write new equations or edit old equations.

- Use DEL to remove the highlighted equation.
- Use CHOOS to add an equation that is already defined in your variables menu, but not listed in the PLOT – FUNCTION window.
- Use ERASE to erase any graph currently existing in the graphics display window.
- Use DRAW to produce the graph according to the current contents of PPAR for the equations listed in the PLOT-FUNCTION window.
- Press NXT to activate the second menu list.

- Use MOVE↑ and MOVE↓ to move the selected equation one location up or down, respectively.
- Use CLEAR if you want to clear all the equations currently active in the PLOT – FUNCTION window. The calculator will verify whether or not you want to clear all the functions before erasing all of them. Select YES, and press OK to proceed with clearing all functions. Select NO, and press OK to de-activate the option CLEAR.
- Press OK when done to return to normal calculator display.

↰ WIN, simultaneously if in RPN mode: Access to the PLOT WINDOW screen.

*Settings*:

- Enter lower and upper limits for horizontal view (H-View) and vertical view (V-View) ranges in the plot window. Or,
- Enter lower and upper limits for horizontal view (H-View), and press AUTO, while the cursor is in one of the V-View fields, to generate the vertical view (V-View) range automatically. Or,
- Enter lower and upper limits for vertical view (V-View), and press AUTO, while the cursor is in one of the H-View fields, to generate the horizontal view (H-View) range automatically.
- The calculator will use the horizontal view (H-View) range to generate data values for the graph, unless you change the options Indep Low, (Indep) High, and (Indep) Step. These values determine, respectively, the minimum, maximum, and increment values of the independent variable to be used in the plot. If the option Default is listed in the fields Indep Low, (Indep) High, and (Indep) Step, the calculator will use the minimum and maximum values determined by H-View.

- A check on _Pixels means that the values of the independent variable increments (Step:) are given in pixels rather than in plot coordinates.

*Soft menu key options*:

- Use EDIT to edit any entry in the window.
- Use AUTO as explained in *Settings*, above.

- Use ERASE to erase any graph currently existing in the graphics display window.
- Use DRAW to produce the graph according to the current contents of PPAR for the equations listed in the PLOT-FUNCTION window.
- Press NXT to activate the second menu list.
- Use RESET to reset the field selected (i.e., where the cursor is positioned) to its default value.
- Use CALC to access calculator stack to perform calculations that may be necessary to obtain a value for one of the options in this window. When the calculator stack is made available to you, you will also have the soft menu key options CANCL and OK .
- Use CANCL in case you want to cancel the current calculation and return to the PLOT WINDOW screen. Or,
- Use OK to accept the results of your calculation and return to the PLOT WINDOW screen.
- Use TYPES to get information on the type of objects that can be used in the selected option field.
- Use CANCL to cancel any changes to the PLOT WINDOW screen and return to normal calculator display.
- Press OK to accept changes to the PLOT WINDOW screen and return to normal calculator display.

← GRAPH, simultaneously if in RPN mode: Plots the graph based on the settings stored in variable PPAR and the current functions defined in the PLOT – FUNCTION screen. If a graph, different from the one you are plotting, already exists in the graphic display screen, the new plot will be superimposed on the existing plot. This may not be the result you desire, therefore, I recommend to use the ERASE DRAW soft menu keys available in the PLOT SETUP, PLOT-FUNCTION or PLOT WINDOW screens.

## Plots of trigonometric and hyperbolic functions

The procedures used above to plot LN(X) and EXP(X), separately or simultaneously, can be used to plot any function of the form y = f(x). It is left as an exercise to the reader to produce the plots of trigonometric and hyperbolic functions and their inverses. The table below suggests the values to use for the vertical and horizontal ranges in each case. You can include the function Y=X when plotting simultaneously a function and its inverse to verify their 'reflection' about the line Y = X.

| | H-View range | | V-View range | |
|---|---|---|---|---|
| **Function** | **Minimum** | **Maximum** | **Minimum** | **Maximum** |
| SIN(X) | -3.15 | 3.15 | AUTO | |
| ASIN(X) | -1.2 | 1.2 | AUTO | |
| SIN & ASIN | -3.2 | 3.2 | -1.6 | 1.6 |
| COS(X) | -3.15 | 3.15 | AUTO | |
| ACOS(X) | -1.2 | 1.2 | AUTO | |
| COS & ACOS | -3.2 | 3.2 | -1.6 | 1.6 |
| TAN(X) | -3.15 | 3.15 | -10 | 10 |
| ATAN(X) | -10 | 10 | -1.8 | 1.8 |
| TAN & ATAN | -2 | -2 | -2 | -2 |
| SINH(X) | -2 | 2 | AUTO | |
| ASINH(X) | -5 | 5 | AUTO | |
| SINH & ASINH | -5 | 5 | -5 | 5 |
| COSH(X) | -2 | 2 | AUTO | |
| ACOSH(X) | -1 | 5 | AUTO | |
| COS & ACOS | -5 | 5 | -1 | 5 |
| TANH(X) | -5 | 5 | AUTO | |
| ATANH(X) | -1.2 | 1.2 | AUTO | |
| TAN & ATAN | -5 | 5 | -2.5 | 2.5 |

## Generating a table of values for a function

The combinations ↰ TBLSET (F5) and ↰ TABLE (F6), pressed simultaneously if in RPN mode, let's the user produce a table of values of functions. For example, we will produce a table of the function Y(X) = X/(X+10), in the range -5 < X < 5 following these instructions:

- We will generate values of the function f(x), defined above, for values of x from –5 to 5, in increments of 0.5. First, we need to ensure that the graph type is set to **FUNCTION** in the PLOT SETUP screen (↰ 2D/3D , press them simultaneously, if in RPN mode). The field in front of the *Type* option will be highlighted. If this field is not already set to **FUNCTION**, press the soft key CHOOS and select the **FUNCTION** option, then press OK.

- Next, press ▼ to highlight the field in front of the option EQ, and type the function expression: ‘X/(X+10)’
- To accept the changes made to the PLOT SETUP screen press NXT OK. You will be returned to normal calculator display.

- The next step is to access the Table Set-up screen by using the keystroke combination ↰ TBLSET (i.e., soft key F5 ) – simultaneously if in RPN mode. This will produce a screen where you can select the starting value (*Start*) and the increment (*Step*). Enter the following: 5 +/- OK 0 • 5 OK 0 • 5 OK (i.e., Zoom factor = 0.5). Toggle the ✓CHK soft menu key until a check mark appears in front of the option *Small Font* if you so desire. Then press OK. This will return you to normal calculator display.

## The TPAR variable

After finishing the table set up, your calculator will create a variable called TPAR (Table PARameters) that store information relevant to the table that is to be generated. To see the contents of this variable, press ↱ TPAR.

- To see the table, press ↰ TABLE (i.e., soft menu key F6 ) – simultaneously if in RPN mode. This will produce a table of values of *x* = *-5, -4.5, …,* and the corresponding values of f(x), listed as Y1 by default. You can use the up and down arrow keys to move about in the table. You will notice that we did not have to indicate an ending value for the independent variable x. Thus, the table continues beyond the maximum value for x suggested early, namely x = 5.

Some options available while the table is visible are ZOOM, BIG, and DEFN:

- The DEFN, when selected, shows the definition of the independent variable.
- The BIG key simply changes the font in the table from small to big, and vice versa. Try it.

- The ZOOM key, when pressed, produces a menu with the options: *In*, *Out*, *Decimal, Integer*, and *Trig*. Try the following exercises:

  - With the option *In* highlighted, press OK. The table is expanded so that the x-increment is now 0.25 rather than 0.5. Simply, what the calculator does is to multiply the original increment, 0.5, by the zoom factor, 0.5, to produce the new increment of 0.25. Thus, the *zoom in* option is useful when you want more resolution for the values of x in your table.
  - To increase the resolution by an additional factor of 0.5 press ZOOM, select *In* once more, and press OK. The x-increment is now 0.0125.
  - To recover the previous x-increment, press ZOOM ▲ OK to select the option *Un-zoom*. The x-increment is increased to 0.25.
  - To recover the original x-increment of 0.5 you can do an *un-zoom* again, or use the *option zoom out* by pressing ZOOM OK.
  - The option Decimal in ZOOM produces x-increments of 0.10.
  - The option Integer in ZOOM produces x-increments of 1.
  - The option Trig in produces increments related to fractions of $\pi$, thus being useful when plotting trigonometric functions.
  - To return to normal calculator display press ENTER.

## Plots in polar coordinates

First of all, you may want to delete the variables used in previous examples (e.g., X, EQ, Y1, PPAR) using function PURGE (TOOL PURGE). By doing this, all parameters related to graphics will be cleared. Press VAR to check that the variables were indeed purged.

We will try to plot the function f(θ) = 2(1-sin(θ)), as follows:

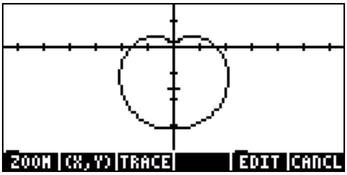
- First, make sure that your calculator's angle measure is set to radians.
- Press ← 2D/3D, simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Polar`, by pressing CHOOS ▼ OK.
- Press ▼ and type:

' 2 × ← () 1 − SIN ALPHA → T OK.

- The cursor is now in the `Indep` field. Press ' ALPHA ↱ T OK to change the independent variable to θ.
- Press NXT OK to return to normal calculator display.
- Press ↰ WIN , simultaneously if in RPN mode, to access the PLOT window (in this case it will be called PLOT –POLAR window).
- Change the H-VIEW range to –8 to 8, by using 8 +/- OK 8 OK, and the V-VIEW range to -6 to 2 by using 6 +/- OK 2 OK.

> **Note**: the H-VIEW and V-VIEW determine the scales of the display window only, and their ranges are not related to the range of values of the independent variable in this case.

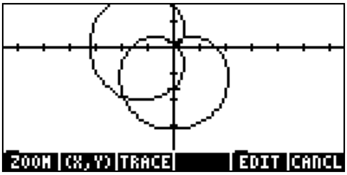
- Change the `Indep Low` value to 0, and the High value to 6.28 (≈ 2π), by using: 0 OK 6 • 2 8 OK.
- Press ERASE DRAW to plot the function in polar coordinates. The result is a curve shaped like a hearth. This curve is known as a *cardiod* (`cardios`, Greek for heart).



- Press EDIT NXT LABEL MENU to see the graph with labels. Press NXT to recover the menu. Press NXT PICT to recover the original graphics menu.
- Press TRACE (X,Y) to trace the curve. The data shown at the bottom of the display is the angle θ and the radius r, although the latter is labeled Y (default name of dependent variable).
- Press NXT CANCL to return to the PLOT WINDOW screen. Press NXT OK to return to normal calculator display.

In this exercise we entered the equation to be plotted directly in the PLOT SETUP window. We can also enter equations for plotting using the PLOT window, i.e., simultaneously if in RPN mode, pressing ↰ Y= . For example, when you press ↰ Y= after finishing the previous exercise, you will get the equation ‘2*(1-SIN(θ))’ highlighted. Let’s say, we want to plot also the function ‘2*(1-COS(θ))’ along with the previous equation.

- Press ADD , and type 2 × ← () 1 − COS ALPHA → T ENTER, to enter the new equation.
- Press ERASE DRAW to see the two equations plotted in the same figure. The result is two intersecting *cardioids*. Press CANCL ON to return to normal calculator display.



## Plotting conic curves

The most general form of a conic curve in the x-y plane is: $Ax^2+By^2+Cxy+Dx+Ey+F = 0$. We also recognize as conic equations those given in the canonical form for the following figures:

- circle: $(x-x_o)^2+(y-y_o)^2 = r^2$
- ellipse: $(x-x_o)^2/a^2 + (y-y_o)^2/b^2 = 1$
- parabola: $(y-b)^2 = K(x-a)$ or $(x-a)^2 = K(y-b)$
- hyperbola: $(x-x_o)^2/a^2 + (y-y_o)^2/b^2 = 1$ or $xy = K$,

where $x_o$, $y_o$, $a$, $b$, and $K$ are constant.

The name *conic curves* follows because these figures (circles, ellipses, parabolas or hyperbolas) result from the intersection of a plane with a cone. For example, a circle is the intersection of a cone with a plane perpendicular to the cone's main axis.

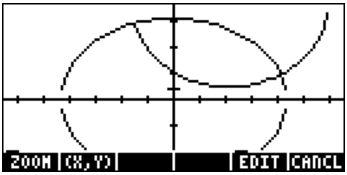
The calculator has the ability of plotting one or more conic curves by selecting Conic as the function TYPE in the PLOT environment. Make sure to delete the variables PPAR and EQ before continuing. For example, let's store the list of equations

{ '(X-1)^2+(Y-2)^2=3' , 'X^2/4+Y^2/3=1' }

into the variable EQ.

These equations we recognize as those of a circle centered at (1,2) with radius √3, and of an ellipse centered at (0,0) with semi-axis lengths *a* = 2 and *b* = √3.

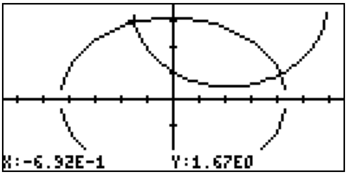
- Enter the PLOT environment, by pressing ↰ 2D/3D , simultaneously if in RPN mode, and select `Conic` as the `TYPE`. The list of equations will be listed in the EQ field.
- Make sure that the independent variable (`Indep`) is set to 'X' and the dependent variable (`Depnd`) to 'Y'.
- Press NXT OK to return to normal calculator display.
- Enter the PLOT WINDOW environment, by pressing ↰ WIN , simultaneously if in RPN mode.
- Change the range for H-VIEW to -3 to 3, by using 3 +/- OK 3 OK. Also, change the V-VIEW range to -1.5 to 2 by using 1 • 5 +/- OK 2 OK.
- Change the *Indep Low*: and *High:* fields to Default by using NXT RESET while each of those fields is highlighted. Select the option *Reset value* after pressing RESET. Press OK to complete the resetting of values. Press NXT to return to the main menu.
- Plot the graph: ERASE DRAW.

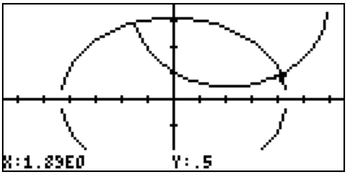


**Note**: The `H-View` and `V-View` ranges were selected to show the intersection of the two curves. There is no general rule to select those ranges, except based on what we know about the curves. For example, for the equations shown above, we know that the circle will extend from -3+1 = -2 to 3+1 = 4 in x, and from -3+2=-1 to 3+2=5 in y. In addition, the ellipse, which is centered at the origin (0,0), will extend from -2 to 2 in x, and from -√3 to √3 in y.

Notice that for the circle and the ellipse the region corresponding to the left and right extremes of the curves are not plotted. This is the case with all circles or ellipses plotted using `Conic` as the `TYPE`.

- To see labels: EDIT NXT LABEL MENU
- To recover the menu: NXT NXT PICT
- To estimate the coordinates of the point of intersection, press the (X,Y) menu key and move the cursor as close as possible to those points using the arrow keys. The coordinates of the cursor are shown in the display. For example, the left point of intersection is close to (-0.692, 1.67), while the right intersection is near (1.89,0.5).





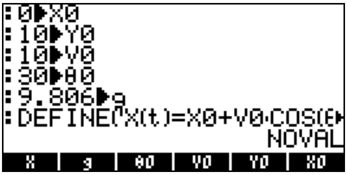
- To recover the menu and return to the PLOT environment, press NXT CANCL.
- To return to normal calculator display, press NXT ON.

## Parametric plots

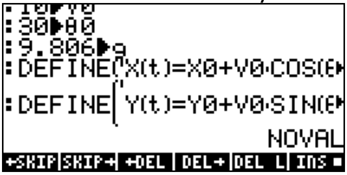
Parametric plots in the plane are those plots whose coordinates are generated through the system of equations x = x(t) and y = y(t), where t is known as the parameter. An example of such graph is the trajectory of a projectile, $x(t) = x_0 + v_0 \cdot \text{COS}\ \theta_0 \cdot t$, $y(t) = y_0 + v_0 \cdot \sin\theta_0 \cdot t - ½ \cdot g \cdot t^2$. To plot equations like these, which involve constant values $x_0$, $y_0$, $v_0$, and $\theta_0$, we need to store the values of those parameters in variables. To develop this example, create a sub-directory called 'PROJM' for PROJectile Motion, and within that sub-directory store the following variables: X0 = 0, Y0 = 10, V0 = 10 , θ0 = 30, and g = 9.806. Make sure that the calculator's angle measure is set to DEG. Next, define the functions (use ← DEF ):

X(t) = X0 + V0*COS(θ0)*t
Y(t) = Y0 + V0*SIN(θ0)*t – 0.5*g*t^2

which will add the variables Y and X to the soft menu key labels.



```
:0▶X0
:10▶Y0
:10▶V0
:30▶θ0
:9.806▶g
:DEFINE('X(t)=X0+V0·COS(θ▶
                     NOVAL
```



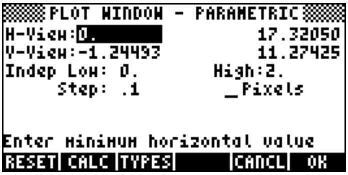
```
:10▶Y0
:30▶θ0
:9.806▶g
:DEFINE('X(t)=X0+V0·COS(θ▶
:DEFINE('Y(t)=Y0+V0·SIN(θ▶
                     NOVAL
```

To produce the graph itself, follow these steps:

- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Parametric`, by pressing CHOOS ▽ ▽ OK.
- Press ▽ and type 'X(t) + i*Y(t)' OK to define the parametric plot as that of a complex variable. (The real and imaginary parts of the complex variable correspond to the x- and y-coordinates of the curve.)
- The cursor is now in the `Indep` field. Press ' ALPHA ← T OK to change the independent variable to *t*.
- Press NXT OK to return to normal calculator display.
- Press ← WIN , simultaneously if in RPN mode, to access the PLOT window (in this case it will be called PLOT –PARAMETRIC window). Instead of modifying the horizontal and vertical views first, as done for other types of plot, we will set the lower and upper values of the independent variable first as follows:
- Select the `Indep Low` field by pressing ▽▽. Change this value to 0 OK. Then, change the value of `High` to 2 OK. Enter 0 • 1 OK for the `Step` value (i.e., step = 0.1).

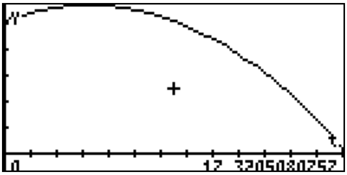
**Note**: Through these settings we are indicating that the parameter t will take values of t = 0, 0.1, 0.2, …, etc., until reaching the value of 2.0.

- Press AUTO. This will generate automatic values of the H-View and V-View ranges based on the values of the independent variable t and the definitions of X(t) and Y(t) used. The result will be:



```
PLOT WINDOW - PARAMETRIC
H-View:0.                17.32050
V-View:-1.24493          11.27425
Indep Low: 0.         High:2.
     Step: .1         _Pixels

Enter minimum horizontal value
RESET CALC TYPES       CANCL  OK
```

- Press ERASE DRAW to draw the parametric plot.
- Press EDIT NXT LABEL MENU to see the graph with labels. The window parameters are such that you only see half of the labels in the x-axis.



- Press NXT to recover the menu. Press NXT PICT to recover the original graphics menu.
- Press TRACE (X,Y) to determine coordinates of any point on the graph. Use ▶ and ◀ to move the cursor about the curve. At the bottom of the screen you will see the value of the parameter t and coordinates of the cursor as (X,Y).
- Press NXT CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

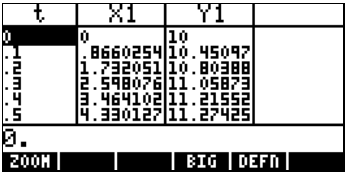
A review of your soft menu key labels shows that you now have the following variables: t, EQ, PPAR, Y, X, g, θ0, V0, Y0, X0. Variables t, EQ, and PPAR are generated by the calculator to store the current values of the parameter, t, of the equation to be plotted EQ (which contains 'X(t) + I∗Y(t)'), and the plot parameters. The other variables contain the values of constants used in the definitions of X(t) and Y(t).

You can store different values in the variables and produce new parametric plots of the projectile equations used in this example. If you want to erase the current picture contents before producing a new plot, you need to access either the PLOT, PLOT WINDOW, or PLOT SETUP screens, by pressing, ← Y= , ← WIN , or ← 2D/3D (the two keys must be pressed simultaneously if in RPN mode). Then, press ERASE DRAW. Press CANCL to return to the PLOT, PLOT WINDOW, or PLOT SETUP screen. Press ON, or NXT OK, to return to normal calculator display.

### Generating a table for parametric equations

In an earlier example we generated a table of values (X,Y) for an expression of the form Y=f(X), i.e., a Function type of graph. In this section, we present the procedure for generating a table corresponding to a parametric plot. For this purpose, we'll take advantage of the parametric equations defined in the example above.

- First, let's access the TABLE SETUP window by pressing ↰ TBLSET, simultaneously if in RPN mode. For the independent variable change the Starting value to 0.0, and the Step value to 0.1. Press OK.
- Generate the table by pressing, simultaneously if in RPN mode, ↰ TABLE. The resulting table has three columns representing the parameter t, and the coordinates of the corresponding points. For this table the coordinates are labeled X1 and Y1.



| t | X1 | Y1 | |
|---|---|---|---|
| 0 | 0 | 10 | |
| .1 | .8660254 | 10.45097 | |
| .2 | 1.732051 | 10.80388 | |
| .3 | 2.598076 | 11.05873 | |
| .4 | 3.464102 | 11.21552 | |
| .5 | 4.330127 | 11.27425 | |

0.

- Use the arrow keys, ◀ ▶ ▲ ▼, to move about the table.
- Press ON to return to normal calculator display.

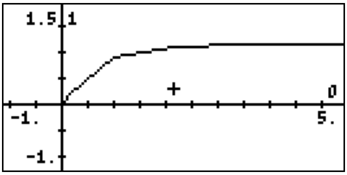
This procedure for creating a table corresponding to the current type of plot can be applied to other plot types.

## Plotting the solution to simple differential equations

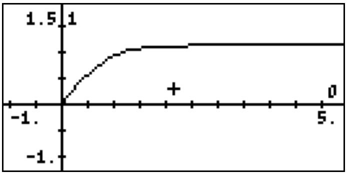
The plot of a simple differential equation can be obtained by selecting `Diff Eq` in the `TYPE` field of the PLOT SETUP environment as follows: suppose that we want to plot x(t) from the differential equation dx/dt = exp($-t^2$), with initial conditions: x = 0 at t = 0. The calculator allows for the plotting of the solution of differential equations of the form Y'(T) = F(T,Y). For our case, we let Y→x and T→t, therefore, F(T,Y)→f(t,x) = exp(-$t^2$).

Before plotting the solution, x(t), for t = 0 to 5, delete the variables EQ and PPAR.

- Press ↰ 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Diff Eq`.
- Press ▼ and type ' ↰ $e^x$ − ALPHA ↰ T $Y^x$ 2 OK.
- The cursor is now in the `H-Var` field. It should show `H-Var:0` and also `V-Var:1`. This is the code used by the calculator to identify the variables to be plotted. `H-Var:0` means the independent variable (to be selected later) will be plotted in the horizontal axis. Also, `V-Var:1` means the dependent variable (default name 'Y') will be plotted in the vertical axis.
- Press ▼ . The cursor is now in the `Indep` field. Press ↱ ' ALPHA ↰ T OK to change the independent variable to *t.*
- Press NXT OK to return to normal calculator display.
- Press ↰ WIN , simultaneously if in RPN mode, to access the PLOT window (in this case it will be called PLOT WINDOW – DIFF EQ).
- Change the H-VIEW and V-VIEW parameters to read: H-VIEW: -1 5, V-VIEW: -1 1.5
- Change the `Init` value to 0, and the Final value to 5 by using: 0 OK 5 OK.
- The values Step and Tol represent the step in the independent variable and the tolerance for convergence to be used by the numerical solution. Let's leave those values with their default settings (if the word *default* is not shown in the Step: field, use NXT RESET to reset that value to its default value. Press NXT to return to the main menu.) Press ▼ .
- The Init-Soln value represents the initial value of the solution to start the numerical result. For the present case, we have for initial conditions x(0) = 0, thus, we need to change this value to 0.0, by using 0 OK.
- Press ERASE DRAW to plot the solution to the differential equation.
- Press EDIT NXT LABEL MENU to see the graph with labels.



- Press NXT to recover the menu. Press NXT PICT to recover the original graphics menu.
- When we observed the graph being plotted, you'll notice that the graph is not very smooth. That is because the plotter is using a time step that is too large. To refine the graph and make it smoother, use a step of 0.1. Try the following keystrokes: CANCL ▼ ▼ ▼ . / OK ERASE DRAW The plot will take longer to be completed, but the shape is definitely smoother than before.
- Press EDIT NXT LABEL MENU, to see axes labels and range. Notice that the labels for the axes are shown as 0 (horizontal) and 1 (vertical). These are the definitions for the axes as given in the PLOT WINDOW screen (see above), i.e., H-VAR (t): 0, and V-VAR(x): 1.



- Press NXT NXT PICT to recover menu and return to PICT environment.
- Press (X,Y) to determine coordinates of any point on the graph. Use ▶ and ◀ to move the cursor in the plot area. At the bottom of the screen you will see the coordinates of the cursor as (X,Y). The calculator uses X and Y as the default names for the horizontal and vertical axes, respectively.
- Press NXT CANCL to return to the PLOT WINDOW environment. Then, press ON to return to normal calculator display.

More details on using graphical solutions of differential equations are presented in Chapter 16.

## Truth plots

Truth plots are used to produce two-dimensional plots of regions that satisfy a certain mathematical condition that can be either true or false. For example, suppose that you want to plot the region for X^2/36 + Y^2/9 < 1, proceed as follows:

- Press ← 2D/3D, simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change TYPE to Truth.
- Press ▼ and type {'(X^2/36+Y^2/9 < 1)','(X^2/16+Y^2/9 > 1)'} OK to define the conditions to be plotted.
- The cursor is now in the `Indep` field. Leave that as 'X' if already set to that variable, or change it to 'X' if needed.
- Press NXT OK to return to normal calculator display.
- Press ← WIN, simultaneously if in RPN mode, to access the PLOT window (in this case it will be called PLOT WINDOW – TRUTH window). Let's keep the default value for the window's ranges: `H-View: -6.5 6.5`, `V-View: -3.1  3.2` (To reset them use NXT RESET (select Reset all) OK NXT).

**Note**: if the window's ranges are not set to default values, the quickest way to reset them is by using NXT RESET (select *Reset all*) OK NXT.

- Press ERASE DRAW to draw the truth plot. Because the calculator samples the entire plotting domain, point by point, it takes a few minutes to produce a truth plot. The present plot should produce a shaded ellipse of semi-axes 6 and 3 (in x and y, respectively), centered at the origin.
- Press EDIT NXT LABEL MENU to see the graph with labels. The window parameters are such that you only see half of the labels in the x-axis. Press NXT to recover the menu. Press NXT PICT to recover the original graphics menu.
- Press (X,Y) to determine coordinates of any point on the graph. Use the arrow keys to move the cursor about the region plotted. At the bottom of the screen you will see the value of the coordinates of the cursor as (X,Y).
- Press NXT CANCL to return to the PLOT WINDOW environment. Then, press ON, or NXT OK, to return to normal calculator display.

You can have more than one condition plotted at the same time if you multiply the conditions. For example, to plot the graph of the points for which $X^2/36 + Y^2/9 < 1$, and $X^2/16 + Y^2/9 > 1$, use the following:

- Press ↰ 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Press ▼ and type '(X^2/36+Y^2/9 < 1)· (X^2/16+Y^2/9 > 1)' OK to define the conditions to be plotted.
- Press ERASE DRAW to draw the truth plot. Again, you have to be patient while the calculator produces the graph. If you want to interrupt the plot, press ON , once. Then press CANCEL .

## Plotting histograms, bar plots, and scatter plots

Histograms, bar plots and scatter plots are used to plot discrete data stored in the reserved variable ΣDAT. This variable is used not only for these types of plots, but also for all kind of statistical applications as will be shown in Chapter 18. As a matter of fact, the use of histogram plots is postponed until we get to that chapter, for the plotting of a histogram requires to perform a grouping of data and a frequency analysis before the actual plot. In this section we will show how to load data in the variable ΣDAT and how to plot bar plots and scatter plots.

We will use the following data for plotting bar plots and scatter plots:

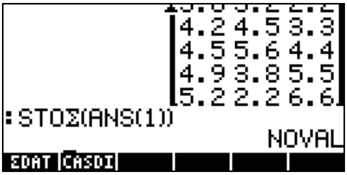
| x | y | z |
|---|---|---|
| 3.1 | 2.1 | 1.1 |
| 3.6 | 3.2 | 2.2 |
| 4.2 | 4.5 | 3.3 |
| 4.5 | 5.6 | 4.4 |
| 4.9 | 3.8 | 5.5 |
| 5.2 | 2.2 | 6.6 |

### Bar plots

First, make sure your calculator's CAS is in `Exact` mode. Next, enter the data shown above as a matrix, i.e.,

[[3.1,2.1,1.1],[3.6,3.2,2.2],[4.2,4.5,3.3],
[4.5,5.6,4.4],[4.9,3.8,5.5],[5.2,2.2,6.6]] ENTER

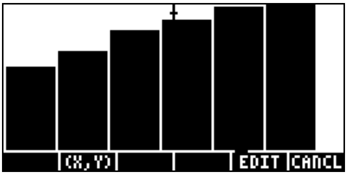
to store it in ΣDAT, use the function STOΣ (available in the function catalog, ‿ CAT ). Press VAR to recover your variables menu. A soft menu key labeled ΣDAT should be available in the stack. The figure below shows the storage of this matrix in ALG mode:



```
                 3.6 3.2 2.2
                [4.2 4.5 3.3
                 4.5 5.6 4.4
                 4.9 3.8 5.5
                 5.2 2.2 6.6]
:STOΣ(ANS(1))
                       NOVAL
```

To produce the graph:

- Press ‿ 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Bar`.
- A matrix will be shown at the ΣDAT field. This is the matrix we stored earlier into ΣDAT.
- Highlight the `Col:` field. This field lets you choose the column of ΣDAT that is to be plotted. The default value is 1. Keep it to plot column 1 in ΣDAT.
- Press NXT OK to return to normal calculator display.
- Press ‿ WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Change the V-View to read, `V-View: 0  5.`
- Press ERASE DRAW to draw the bar plot.

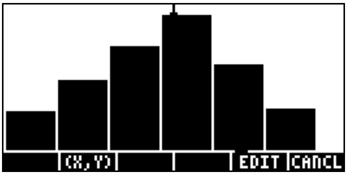


- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

The number of bars to be plotted determines the width of the bar. The H- and V-VIEW are set to 10, by default. We changed the V-VIEW to better accommodate the maximum value in column 1 of ΣDAT. Bar plots are useful when plotting categorical (i.e., non-numerical) data.

Suppose that you want to plot the data in column 2 of the ΣDAT matrix:

- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Press ▼▼ to highlight the `Col:` field and type 2 OK, followed by NXT OK.
- Press ← WIN , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change V-View to read `V-View: 0  6`
- Press ERASE DRAW.



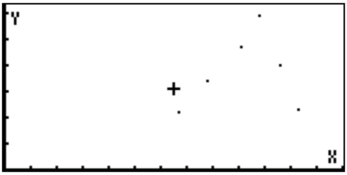
- Press CANCL to return to the PLOT WINDOW screen, then ON to return to normal calculator display.

## Scatter plots

We will use the same ΣDAT matrix to produce scatter plots. First, we will plot the values of y vs. x, then those of y vs. z, as follows:

- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Scatter`.

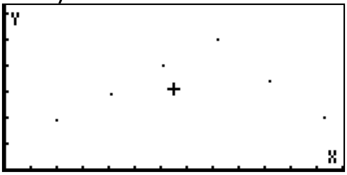
- Press ▼▼ to highlight the **Cols**: field. Enter 1 OK 2 OK to select column 1 as X and column 2 as Y in the Y-vs.-X scatter plot.
- Press NXT OK to return to normal calculator display.
- Press ↰ WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Change the plot window ranges to read: H-View: 0 6, V-View: 0 6.
- Press ERASE DRAW to draw the bar plot. Press EDIT NXT LABEL MENU to see the plot unencumbered by the menu and with identifying labels (the cursor will be in the middle of the plot, however):



- Press NXT NXT EDIT to leave the EDIT environment.
- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

To plot y vs. z, use:

- Press ↰ 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Press ▼▼ to highlight the `Cols`: field. Enter 3 OK 2 OK to select column 3 as X and column 2 as Y in the Y-vs.-X scatter plot.
- Press NXT OK to return to normal calculator display.
- Press ↰ WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Change the plot window ranges to read: H-View: 0 7, V-View: 0 7.
- Press ERASE DRAW to draw the bar plot. Press EDIT NXT LABEL MENU to see the plot unencumbered by the menu and with identifying labels.



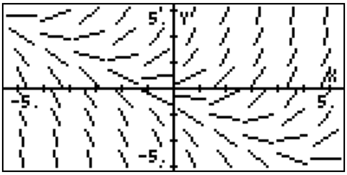
- Press NXT NXT PICT to leave the EDIT environment.
- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

## Slope fields

Slope fields are used to visualize the solutions to a differential equation of the form y' = f(x,y). Basically, what is presented in the plot are segments tangential to the solution curves, since y' = dy/dx, evaluated at any point (x,y), represents the slope of the tangent line at point (x,y).

For example, to visualize the solution to the differential equation y' = f(x,y) = x+y, use the following:

- Press ↰ 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Slopefield`.
- Press ▼ and type 'X+Y' OK.
- Make sure that 'X' is selected as the `Indep:` and 'Y' as the `Depnd:` variables.
- Press NXT OK to return to normal calculator display.
- Press ↰ WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Change the plot window ranges to read: X-Left:-5, X-Right:5, Y-Near:-5, Y-Far: 5
- Press ERASE DRAW to draw the slope field plot. Press EDIT NXT LABEL MENU to see the plot unencumbered by the menu and with identifying labels.



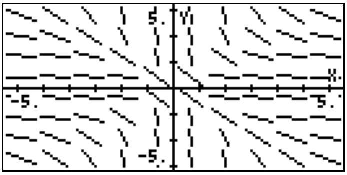
- Press NXT NXT PICT to leave the EDIT environment.

- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

If you could reproduce the slope field plot in paper, you can trace by hand lines that are tangent to the line segments shown in the plot. This lines constitute lines of y(x,y) = constant, for the solution of y' = f(x,y). Thus, slope fields are useful tools for visualizing particularly difficult equations to solve.

Try also a slope field plot for the function y' = f(x,y) = - $(y/x)^2$, by using:

- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Slopefield.`
- Press ▼ and type '– (Y/X)^2' OK.
- Press ERASE DRAW to draw the slope field plot. Press EDIT NXT LABEL MENU to see the plot unencumbered by the menu and with identifying labels.



- Press NXT NXT EXIT to leave the EDIT environment.
- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

## Fast 3D plots

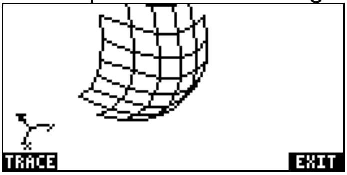
Fast 3D plots are used to visualize three-dimensional surfaces represented by equations of the form z = f(x,y). For example, if you want to visualize z = f(x,y) = $x^2+y^2$, we can use the following:

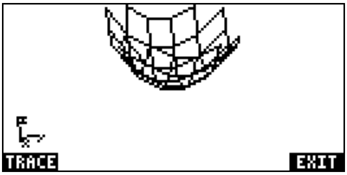
- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change TYPE to Fast3D.
- Press ▼ and type 'X^2+Y^2' OK.

- Make sure that ‘X’ is selected as the Indep: and ‘Y’ as the Depnd: variables.
- Press NXT OK to return to normal calculator display.
- Press ↰ WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Keep the default plot window ranges to read: X-Left:-1, X-Right:1, Y-Near:-1, Y-Far: 1, Z-Low: -1, Z-High: 1, Step Indep: 10, Depnd: 8

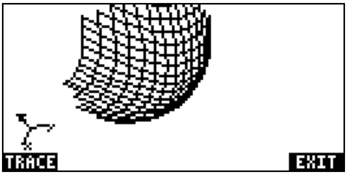
> **Note**: The Step Indep: and Depnd: values represent the number of gridlines to be used in the plot. The larger these number, the slower it is to produce the graph, although, the times utilized for graphic generation are relatively fast. For the time being we’ll keep the default values of 10 and 8 for the Step data.

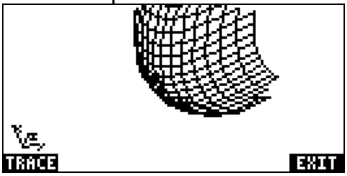
- Press ERASE DRAW to draw the three-dimensional surface. The result is a wireframe picture of the surface with the reference coordinate system shown at the lower left corner of the screen. By using the arrow keys (◀▶▲▼) you can change the orientation of the surface. The orientation of the reference coordinate system will change accordingly. Try changing the surface orientation on your own. The following figures show a couple of views of the graph:





- When done, press EXIT.
- Press CANCL to return to the PLOT WINDOW environment.
- Change the Step data to read: `Step Indep: 20  Depnd: 16`
- Press ERASE DRAW to see the surface plot. Sample views:





- When done, press EXIT.
- Press CANCL to return to PLOT WINDOW.
- Press ON , or NXT OK, to return to normal calculator display.

Try also a Fast 3D plot for the surface z = f(x,y) = sin $(x^2+y^2)$

- Press ← 2D/3D , simultaneously if in RPN mode, to access the PLOT SETUP window.
- Press ▼ and type 'SIN(X^2+Y^2)' OK.
- Press ERASE DRAW to draw the slope field plot. Press EDIT NXT LABEL MENU to see the plot unencumbered by the menu and with identifying labels.
- Press NXT NXT PICT to leave the EDIT environment.
- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

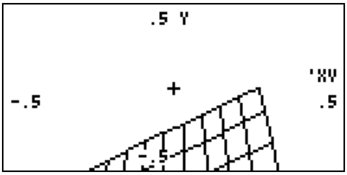
## Wireframe plots

*Wireframe* plots are plots of three-dimensional surfaces described by z = f(x,y). Unlike *Fast 3D* plots, wireframe plots are static plots. The user can choose the viewpoint for the plot, i.e., the point from which the surface is seen. For example, to produce a wireframe plot for the surface z = x + 2y –3, use the following:

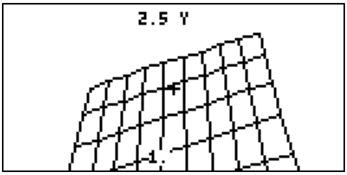
- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Wireframe`.
- Press ▼ and type 'X+2*Y-3' OK.
- Make sure that 'X' is selected as the `Indep:` and 'Y' as the `Depnd:` variables.
- Press NXT OK to return to normal calculator display.
- Press ← WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Keep the default plot window ranges to read: X-Left:-1, X-Right:1, Y-Near:-1, Y-Far: 1, Z-Low: -1, Z-High: 1, XE:0,YE:-3, ZE:0, Step Indep: 10 Depnd: 8

The coordinates XE, YE, ZE, stand for “eye coordinates,” i.e., the coordinates from which an observer sees the plot. The values shown are the default values. The Step Indep: and Depnd: values represent the number of gridlines to be used in the plot. The larger these number, the slower it is to produce the graph. For the time being we’ll keep the default values of 10 and 8 for the Step data.

- Press ERASE DRAW to draw the three-dimensional surface. The result is a wireframe picture of the surface.
- Press EDIT NXT LABEL MENU to see the graph with labels and ranges. This particular version of the graph is limited to the lower part of the display. We can change the viewpoint to see a different version of the graph.



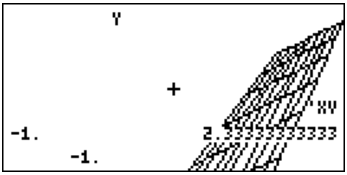
- Press NXT NXT PLOT CANCL to return to the PLOT WINDOW environment.
- Change the eye coordinate data to read : `XE:0    YE:-3    ZE:3`
- Press ERASE DRAW to see the surface plot.
- Press EDIT NXT LABEL MENU to see the graph with labels and ranges.



This version of the graph occupies more area in the display than the previous one. We can change the viewpoint, once more, to see another version of the graph.

- Press NXT NXT PLOT CANCL to return to the PLOT WINDOW environment.
- Change the eye coordinate data to read : `XE:3    YE:3    ZE:3`

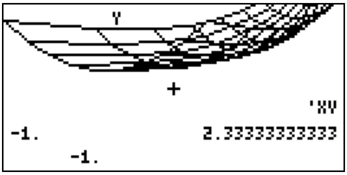
- Press ERASE DRAW to see the surface plot. This time the bulk of the plot is located towards the right –hand side of the display.



- Press CANCL to return to the PLOT WINDOW environment.
- Press ON , or NXT OK, to return to normal calculator display.

Try also a Wireframe plot for the surface z = f(x,y) = $x^2+y^2$

- Press ↰ 2D/3D , simultaneously if in RPN mode, to access the PLOT SETUP window.
- Press ▼ and type 'X^2+Y^2' OK.
- Press ERASE DRAW to draw the slope field plot. Press EDIT NXT MENU LABEL to see the plot unencumbered by the menu and with identifying labels.

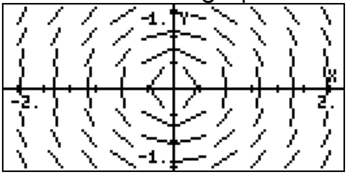


- Press NXT NXT EXIT to leave the EDIT environment.
- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

## Ps-Contour plots

*Ps-Contour* plots are contour plots of three-dimensional surfaces described by z = f(x,y). The contours produced are projections of level surfaces z = constant on the x-y plane. For example, to produce a Ps-Contour plot for the surface z = $x^2+y^2$, use the following:

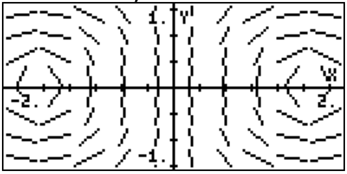
- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change TYPE to Ps-Contour.
- Press ▼ and type 'X^2+Y^2' OK.
- Make sure that 'X' is selected as the `Indep:` and 'Y' as the `Depnd:` variables.
- Press NXT OK to return to normal calculator display.
- Press ← WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Keep the default plot window ranges to read: X-Left:-2, X-Right:2, Y-Near:-1 Y-Far: 1, Step Indep: 10, Depnd: 8
- Press ERASE DRAW to draw the contour plot. This operation will take some time, so, be patient. The result is a contour plot of the surface. Notice that the contour are not necessarily continuous, however, they do provide a good picture of the level surfaces of the function.
- Press EDIT NXT LABEL MENU to see the graph with labels and ranges.



- Press NXT NXT PICT CANCL to return to the PLOT WINDOW environment.
- Press ON , or NXT OK, to return to normal calculator display.

Try also a Ps-Contour plot for the surface z = f(x,y) = sin x cos y.

- Press ← 2D/3D , simultaneously if in RPN mode, to access the PLOT SETUP window.
- Press ▼ and type 'SIN(X)*COS(Y)' OK.
- Press ERASE DRAW to draw the slope field plot. Press EDIT NXT LABEL MENU to see the plot unencumbered by the menu and with identifying labels.

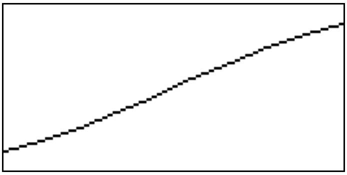


- Press NXT NXT EDIT to leave the EDIT environment.
- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

## Y-Slice plots

*Y-Slice* plots are animated plots of z-vs.-y for different values of x from the function z = f(x,y). For example, to produce a Y-Slice plot for the surface z = $x^3$-$xy^3$, use the following:

- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change TYPE to Y-Slice.
- Press ▼ and type 'X^3+X*Y^3' OK.
- Make sure that 'X' is selected as the Indep: and 'Y' as the Depnd: variables.
- Press NXT OK to return to normal calculator display.
- Press ← WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Keep the default plot window ranges to read: X-Left:-1, X-Right:1, Y-Near:-1, Y-Far: 1, Z-Low:-1, Z-High:1, Step Indep: 10 Depnd: 8
- Press ERASE DRAW to draw the three-dimensional surface. You will see the calculator produce a series of curves on the screen, that will immediately disappear. When the calculator finishes producing all the y-slice curves, then it will automatically go into animating the different curves. One of the curves is shown below.



- Press ON to stop the animation. Press CANCL to return to the PLOT WINDOW environment.
- Press ON , or NXT OK, to return to normal calculator display.

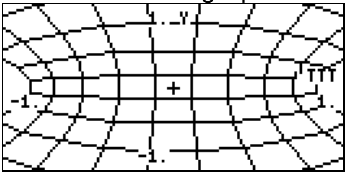
Try also a Ps-Contour plot for the surface z = f(x,y) = (x+y) sin y.

- Press ← 2D/3D , simultaneously if in RPN mode, to access the PLOT SETUP window.
- Press ▼ and type '(X+Y)*SIN(Y)' OK.
- Press ERASE DRAW to produce the Y-Slice animation.
- Press ON to stop the animation.
- Press CANCL to return to the PLOT WINDOW environment. Then, press ON , or NXT OK, to return to normal calculator display.

## Gridmap plots

*Gridmap* plots produce a grid of orthogonal curves describing a function of a complex variable of the form w =f(z) = f(x+iy), where z = x+iy is a complex variable. The functions plotted correspond to the real and imaginary part of w = Φ(x,y) + iΨ(x,y), i.e., they represent curves Φ(x,y) =constant, and Ψ(x,y) = constant. For example, to produce a Gridmap plot for the function w = sin(z), use the following:

- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change TYPE to Gridmap.
- Press ▼ and type 'SIN(X+i*Y)' OK.
- Make sure that 'X' is selected as the Indep: and 'Y' as the Depnd: variables.
- Press NXT OK to return to normal calculator display.
- Press ← WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Keep the default plot window ranges to read: X-Left:-1, X-Right:1, Y-Near:-1 Y-Far: 1, XXLeft:-1 XXRight:1, YYNear:-1, yyFar: 1, Step Indep: 10 Depnd: 8
- Press ERASE DRAW to draw the gridmap plot. The result is a grid of functions corresponding to the real and imaginary parts of the complex function.
- Press EDIT NXT LABEL MENU to see the graph with labels and ranges.



- Press NXT NXT PICT CANCL to return to the PLOT WINDOW environment.

- Press ON , or NXT OK, to return to normal calculator display.

Other functions of a complex variable worth trying for Gridmap plots are:

| | | | |
|---|---|---|---|
| (1) SIN((X,Y)) | i.e., F(z) = sin(z) | (2)(X,Y)^2 | i.e., F(z) = $z^2$ |
| (3) EXP((X,Y)) | i.e., F(z) = $e^z$ | (4) SINH((X,Y)) | i.e., F(z) = sinh(z) |
| (5) TAN((X,Y)) | i.e., F(z) = tan(z) | (6) ATAN((X,Y)) | i.e., F(z) = $\tan^{-1}(z)$ |
| (7) (X,Y)^3 | i.e., F(z) = $z^3$ | (8) 1/(X,Y) | i.e., F(z) = 1/z |
| (9) √ (X,Y) | i.e., F(z) = $z^{1/2}$ | | |

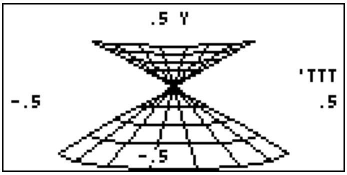
# Pr-Surface plots

*Pr-Surface* (parametric surface) plots are used to plot a three-dimensional surface whose coordinates (x,y,z) are described by x = x(X,Y), y = y(X,Y), z=z(X,Y), where X and Y are independent parameters.

**Note**: The equations x = x(X,Y), y = y(X,Y), z=z(X,Y) represent a parametric description of a surface. X and Y are the independent parameters. Most textbooks will use (u,v) as the parameters, rather than (X,Y). Thus, the parametric description of a surface is given as x = x(u,v), y = y(u,v), z=z(u,v).

For example, to produce a Pr-Surface plot for the surface x = x(X,Y) = X sin Y, y = y(X,Y) = x cos Y, z=z(X,Y)=X, use the following:

- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change TYPE to Pr-Surface.
- Press ▼ and type '{X*SIN(Y), X*COS(Y), X}' OK.
- Make sure that 'X' is selected as the Indep: and 'Y' as the Depnd: variables.
- Press NXT OK to return to normal calculator display.
- Press ← WIN , simultaneously if in RPN mode, to access the PLOT WINDOW screen.
- Keep the default plot window ranges to read: X-Left:-1, X-Right:1, Y-Near:-1, Y-Far: 1, Z-Low: -1, Z-High:1, XE: 0, YE:-3, zE:0, Step Indep: 10, Depnd: 8
- Press ERASE DRAW to draw the three-dimensional surface.
- Press EDIT NXT LABEL MENU to see the graph with labels and ranges.



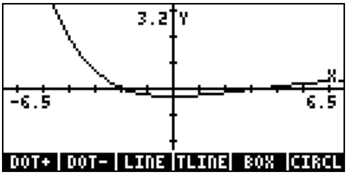
- Press NXT NXT PICT CANCL to return to the PLOT WINDOW environment.
- Press ON , or NXT OK, to return to normal calculator display.

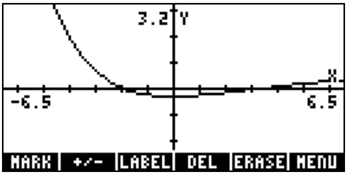
## The VPAR variable

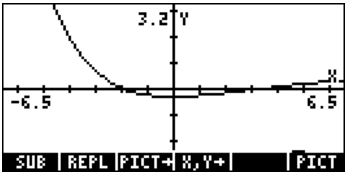
The VPAR (Volume Parameter) variable contains information regarding the "volume" used to produce a three dimensional graph. Therefore, you will see it produced whenever you create a three dimensional plot such as Fast3D, Wireframe, or Pr-Surface.

# Interactive drawing

Whenever we produce a two-dimensional graph, we find in the graphics screen a soft menu key labeled EDIT. Pressing EDIT produces a menu that include the following options (press NXT to see additional functions):







Through the examples above, you have the opportunity to try out functions LABEL, MENU, PICT→, and REPL. Many of the remaining functions, such as DOT+, DOT-, LINE, BOX, CIRCL, MARK, DEL, etc., can be used to draw points, lines, circles, etc. on the graphics screen, as described below. To see how to use these functions we will try the following exercise:

First, we get the graphics screen corresponding to the following instructions:

- Press ← 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `Function`, if needed
- Change EQ to 'X'
- Make sure that Indep: is set to 'X' also
- Press NXT OK to return to normal calculator display.
- Press ← WIN , simultaneously if in RPN mode, to access the PLOT window (in this case it will be called PLOT –POLAR window).
- Change the H-VIEW range to –10 to 10, by using 1 0 +/- OK 1 0 OK, and the V-VIEW range to -5 to 5 by using 5 +/- OK 5 OK.
- Press ERASE DRAW to plot the function.
- Press EDIT NXT LABEL to add labels to the graph. Press NXT NXT (or ← PREV ) to recover the original EDIT menu.

Next, we illustrate the use of the different drawing functions on the resulting graphics screen. They require use of the cursor and the arrow keys (◄ ► ▲ ▼) to move the cursor about the graphics screen.

## DOT+ and DOT-

When DOT+ is selected, pixels will be activated wherever the cursor moves leaving behind a trace of the cursor position. When DOT- is selected, the opposite effect occurs, i.e., as you move the cursor, pixels will be deleted.

For example, use the ► ▲ keys to move the cursor somewhere in the middle of the first quadrant of the x-y plane, then press DOT+. The label will be selected (DOT+■). Press and hold the ► key to see a horizontal line being traced. Now, press DOT-, to select this option ( DOT-■ ). Press and hold the ◄ key to see the line you just traced being erased. Press DOT-, when done, to deselect this option.

## MARK

This command allows the user to set a mark point which can be used for a number of purposes, such as:

- Start of line with the LINE or TLINE command
- Corner for a BOX command
- Center for a CIRCLE command

Using the MARK command by itself simply leaves an x in the location of the mark (Press NXT MARK to see it in action).

## LINE

This command is used to draw a line between two points in the graph. To see it in action, position the cursor somewhere in the first quadrant, and press ↰ PREV LINE. A MARK is placed over the cursor indicating the origin of the line. Use the ▶ key to move the cursor to the right of the current position, say about 1 cm to the right, and press LINE. A line is draw between the first and the last points.

Notice that the cursor at the end of this line is still active indicating that the calculator is ready to plot a line starting at that point. Press ▼ to move the cursor downwards, say about another cm, and press LINE again. Now you should have a straight angle traced by a horizontal and a vertical segments. The cursor is still active. To deactivate it, without moving it at all, press LINE. The cursor returns to its normal shape (a cross) and the LINE function is no longer active.

## TLINE

(Toggle LINE) Move the cursor to the second quadrant to see this function in action. Press TLINE. A MARK is placed at the start of the toggle line. Move the cursor with the arrow keys away from this point, and press TLINE. A line is drawn from the current cursor position to the reference point selected earlier. Pixels that are on in the line path will be turned off, and vice versa. To remove the most recent line traced, press TLINE again. To deactivate TLINE, move the cursor to the original point where TLINE was activated, and press LINE LINE.

## BOX

This command is used to draw a box in the graph. Move the cursor to a clear area of the graph, and press BOX. This highlights the cursor. Move the cursor with the arrow keys to a point away, and in a diagonal direction, from the current cursor position. Press BOX again. A rectangle is drawn whose diagonal joins the initial and ending cursor positions. The initial position of the box is still marked with and x. Moving the cursor to another position and pressing BOX will generate a new box containing the initial point. To deselect BOX, move the cursor to the original point where BOX was activated, then press LINE LINE.

## CIRCL

This command produces a circle. Mark the center of the circle with a MARK command, then move the cursor to a point that will be part of the periphery of the circle, and press CIRCL. To deactivate CIRCL, return the cursor to the MARK position and press LINE.

Try this command by moving the cursor to a clear part of the graph, press MARK. Move the cursor to another point, then press CIRCL. A circle centered at the MARK, and passing through the last point will be drawn.

## LABEL

Pressing LABEL places the labels in the x- and y-axes of the current plot. This feature has been used extensively through this chapter.

## DEL

This command is used to remove parts of the graph between two MARK positions. Move the cursor to a point in the graph, and press MARK. Move the cursor to a different point, press MARK again. Then, press DEL. The section of the graph boxed between the two marks will be deleted.

## ERASE

The function ERASE clears the entire graphics window. This command is available in the PLOT menu, as well as in the plotting windows accessible through the soft menu keys.

## MENU

Pressing MENU will remove the soft key menu labels to show the graphic unencumbered by those labels. To recover the labels, press NXT.

## SUB

Use this command to extract a subset of a graphics object. The extracted object is automatically placed in the stack. Select the subset you want to extract by placing a MARK at a point in the graph, moving the cursor to the diagonal corner of the rectangle enclosing the graphics subset, and press SUB. This feature can be used to move parts of a graphics object around the graph.

## REPL

This command places the contents of a graphic object currently in stack level 1 at the cursor location in the graphics window. The upper left corner of the graphic object being inserted in the graph will be placed at the cursor position. Thus, if you want a graph from the stack to completely fill the graphic window, make sure that the cursor is placed at the upper left corner of the display.

## PICT→

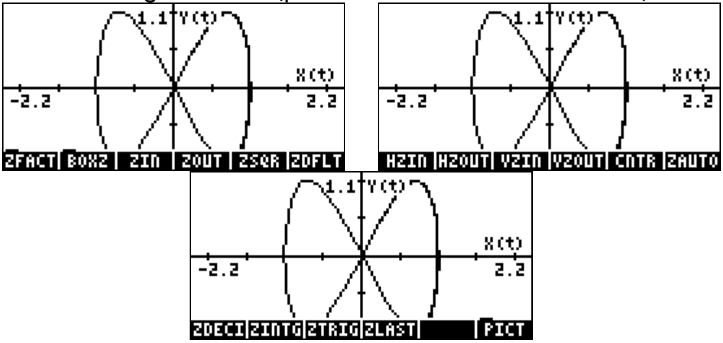
This command places a copy of the graph currently in the graphics window on to the stack as a graphic object. The graphic object placed in the stack can be saved into a variable name for storage or other type of manipulation.

## X,Y→

This command copies the coordinates of the current cursor position, in user coordinates, in the stack.

# Zooming in and out in the graphics display

Whenever you produce a two-dimensional FUNCTION graphic interactively, the first soft-menu key, labeled ZOOM, lets you access functions that can be used to zoom in and out in the current graphics display. The ZOOM menu includes the following functions (press to move to the next menu):


1.1
Y(t)
X(t)
-2.2
2.2
ZFACT BOXZ ZIN ZOUT ZSQR ZDFLT
HZIN HZOUT VZIN VZOUT CNTR ZAUTO
ZDECI ZINTG ZTRIG ZLAST PICT

We present each of these functions following. You just need to produce a graph as indicated in Chapter 12, or with one of the programs listed earlier in this Chapter.

## ZFACT, ZIN, ZOUT, and ZLAST

Pressing ZFACT produces an input screen that allows you to change the current X- and Y-Factors. The X- and Y-Factors relate the horizontal and vertical user-defined unit ranges to their corresponding pixel ranges. Change the H-Factor to read 8., and press OK, then change the V-Factor to read 2., and press OK. Check off the option ✓`Recenter on cursor`, and press OK.

Back in the graphics display, press ZIN. The graphic is re-drawn with the new vertical and horizontal scale factors, centered at the position where the cursor was located, while maintaining the original PICT size (i.e., the original number of pixels in both directions). Using the arrow keys, scroll horizontally or vertically as far as you can of the zoomed-in graph.

To zoom out, subjected to the H- and V-Factors set with ZFACT, press ZOOM ZOUT. The resulting graph will provide more detail than the zoomed-in graph. You can always return to the very last zoom window by using ZLAST.

## BOXZ

Zooming in and out of a given graph can be performed by using the soft-menu key BOXZ. With BOXZ you select the rectangular sector (the "box") that you want to zoom in into. Move the cursor to one of the corners of the box (using the arrow keys), and press ZOOM BOXZ. Using the arrow keys once more, move the cursor to the opposite corner of the desired zoom box. The cursor will trace the zoom box in the screen. When desired zoom box is selected, press ZOOM. The calculator will zoom in the contents of the zoom box that you selected to fill the entire screen.

If you now press ZOUT, the calculator will zoom out of the current box using the H- and V-Factors, which may not recover the graph view from which you started the zoom box operation.

## ZDFLT, ZAUTO

Pressing ZDFLT re-draws the current plot using the default x- and y-ranges, i.e., -6.5 to 6.5 in x, and –3.1 to 3.1 in y. The command ZAUTO, on the other hand, creates a zoom window using the current independent variable (x) range, but adjusting the dependent variable (y) range to fit the curve (as when you use the function AUTO in the PLOT WINDOW input form („ò, simultaneously in RPN mode).

## HZIN, HZOUT, VZIN and VZOUT

These functions zoom in and out the graphics screen in the horizontal or vertical direction according to the current H- and V-Factors.

## CNTR

Zooms in with the center of the zoom window in the current cursor location. The zooming factors used are the current H- and V-Factors.

## ZDECI

Zooms the graph so as to round off the limits of the x-interval to a decimal value.

## ZINTG

Zooms the graph so that the pixel units become user-define units. For example, the minimum PICT window has 131 pixels. When you use ZINTG, with the cursor at the center of the screen, the window gets zoomed so that the x-axis extends from –64.5 to 65.5.

## ZSQR

Zooms the graph so that the plotting scale is maintained at 1:1 by adjusting the x scale, keeping the y scale fixed, if the window is wider than taller. This forces a proportional zooming.

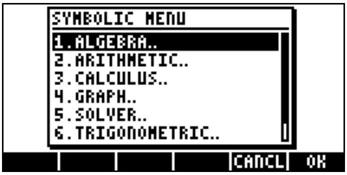
## ZTRIG

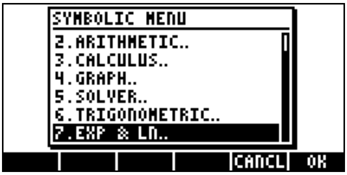
Zooms the graph so that the x scale incorporates a range from about –3π to +3π, the preferred range for trigonometric functions.

**Note**: None of these functions are programmable. They are only useful in an interactive way. Do not confuse the command ZFACT in the ZOOM menu with the function ZFACTOR, which is used for gas dynamic and chemistry applications (see Chapter 3).

# The SYMBOLIC menu and graphs

The SYMBOLIC menu is activated by pressing the SYMB key (fourth key from the left in fourth row from the top of the keyboard). This menu provides a list of menus related to the Computer Algebraic System or CAS, these are:



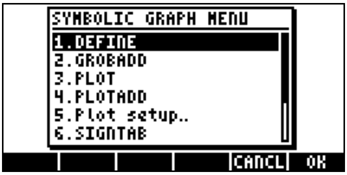


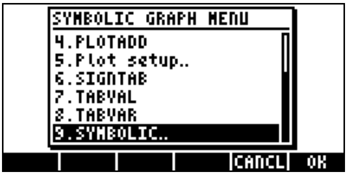
All but one of these menus are available directly in the keyboard by pressing the appropriate keystroke combination as follows. The Chapter of the user manual where the menus are described is also listed:

| | | |
|---|---|---|
| ALGEBRA.. | ↱ ALG (the 4 key) | Ch. 5 |
| ARITHMETIC.. | ↰ ARITH (the 1 key) | Ch. 5 |
| CALCULUS.. | ↰ CALC (the 4 key) | Ch. 13 |
| SOLVER.. | ↰ S.SLV (the 7 key) | Ch. 6 |
| TRIGONOMETRIC.. | ↱ TRIG (the 8 key) | Ch. 5 |
| EXP&LN.. | ↰ EXP&LN (the 8 key) | Ch. 5 |

## The SYMB/GRAPH menu

The GRAPH sub-menu within the SYMB menu includes the following functions:

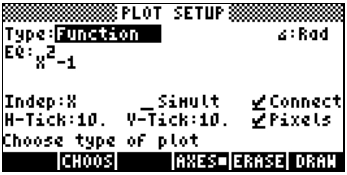


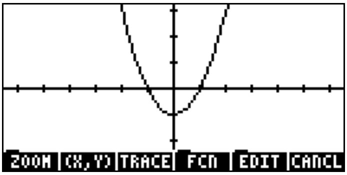


DEFINE: same as the keystroke sequence ↰ DEF (the 2 key)
GROBADD: pastes two GROBs first over the second (See Chapter 22)
PLOT(function): plots a function, similar to ↰ 2D/3D
PLOTADD(function): adds this function to the list of functions to plot, similar to ↰ 2D/3D
Plot setup..: same as ↰ 2D/3D
SIGNTAB(function): sign table of given function showing intervals of positive and negative variation, zero points and infinite asymptotes
TABVAL: table of values for a function
TABVAR: variation table of a function

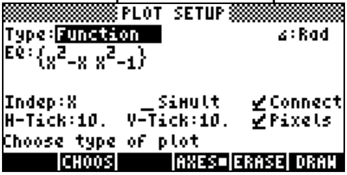
Examples of some of these functions are provided next.

PLOT(X^2-1) is similar to ↰ 2D/3D with EQ: X^2 -1. Using ERASE DRAW produces the plot:

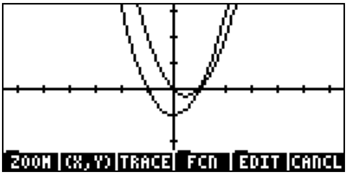

PLOT SETUP
Type: Function ∡:Rad
EQ: $X^2-1$
Indep:X _Simult ✓Connect
H-Tick:10. V-Tick:10. ✓Pixels
Choose type of plot



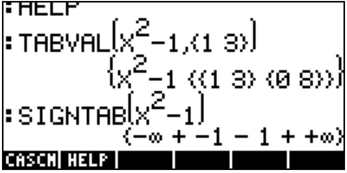
PLOTADD(X^2-X) is similar to ↰ 2D/3D but adding this function to EQ: X^2 -1. Using ERASE DRAW produces the plot:



```
        PLOT SETUP
Type:Function          ∡:Rad
EQ:{X²-X X²-1}

Indep:X      _Simult   ✓Connect
H-Tick:10.  V-Tick:10. ✓Pixels
Choose type of plot
```

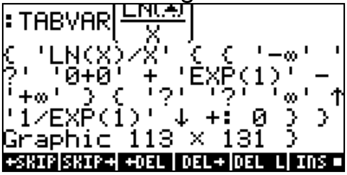


TABVAL(X^2-1,{1, 3}) produces a list of {min max} values of the function in the interval {1,3}, while SIGNTAB(X^2-1) shows the sign of the function in the interval (-∞,+), with f(x) > 0 in (-∞,-1), f(x) <0, in (-1,1), and f(x) > 0 in (1,+ ∞).



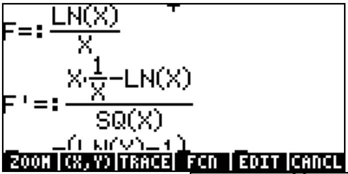
```
:HELP
:TABVAL(X²-1,{1 3})
        {X²-1 {{1 3} {0 8}}}
:SIGNTAB(X²-1)
        {-∞ + -1 - 1 + +∞}
```

TABVAR(LN(X)/X) produces the following table of variation:



```
:TABVAR(LN(X)/X)
{ 'LN(X)/X' { { '-∞' '
?' '0+0' + 'EXP(1)' -
'+∞' } { '?' '?' '∞' ↑
'1/EXP(1)' ↓ +: 0 } }
Graphic 113 × 131 }
```

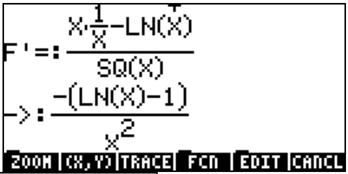
A detailed interpretation of the table of variation is easier to follow in RPN mode:



$$F=:\frac{LN(X)}{X}$$

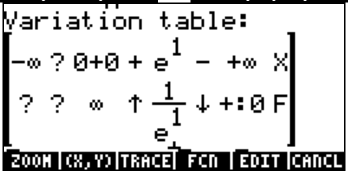
$$F'=:\frac{X\cdot\frac{1}{X}-LN(X)}{SQ(X)}$$

$$-(LN(X)-1)$$



$$F'=:\frac{X\cdot\frac{1}{X}-LN(X)}{SQ(X)}$$

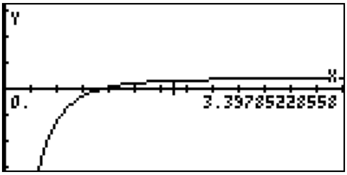
$$->:\frac{-(LN(X)-1)}{X^2}$$



Variation table:

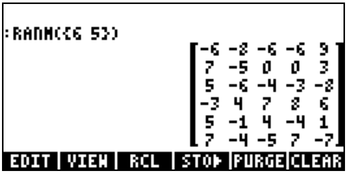
$$\begin{bmatrix} -\infty & ? & 0+0 & + & e^1 & - & +\infty & X \\ ? & ? & \infty & \uparrow & \frac{1}{e^1} & \downarrow & +:0 & F \end{bmatrix}$$

The output is in a graphical format, showing the original function, F(X), the derivative F'(X) right after derivation and after simplification, and finally a table of variation. The table consists of two rows, labeled in the right-hand side. Thus, the top row represents values of X and the second row represents values of F. The question marks indicates uncertainty or non-definition. For example, for X<0, LN(X) is not defined, thus the X lines shows a question mark in that interval. Right at zero (0+0) F is infinite, for X = e, F = 1/e. F increases before reaching this value, as indicated by the upward arrow, and decreases after this value (X=e) becoming slightly larger than zero (+:0) as X goes to infinity. A plot of the graph is shown below to illustrate these observations:

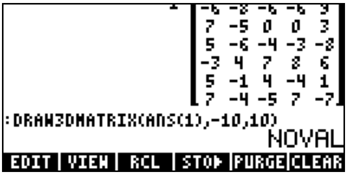


## Function DRAW3DMATRIX

This function takes as argument a n×m matrix, **Z**, = [ $z_{ij}$ ], and minimum and maximum values for the plot. You want to select the values of $v_{min}$ and $v_{max}$ so that they contain the values listed in **Z**. The general call to the function is, therefore, DRAW3DMATRIX(**Z**,$v_{min}$,$v_{max}$). To illustrate the use of this function we first generate a 6×5 matrix using RANM({6,5}), and then call function DRAW3DMATRIX, as shown below:

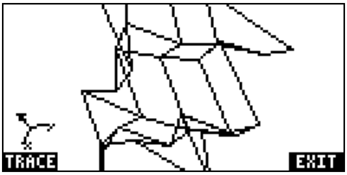


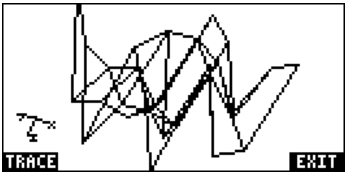
```
:RANM({6 5})
                         [-6 -8 -6 -6  9]
                         [ 7 -5  0  0  3]
                         [ 5 -6 -4 -3 -8]
                         [-3  4  7  8  6]
                         [ 5 -1  4 -4  1]
                         [ 7 -4 -5  7 -7]
```



```
                         -6 -8 -6 -6  9
                          7 -5  0  0  3
                          5 -6 -4 -3 -8
                         -3  4  7  8  6
                          5 -1  4 -4  1
                          7 -4 -5  7 -7]
:DRAW3DMATRIX(ANS(1),-10,10)
                              NOVAL
```

The plot is in the style of a FAST3DPLOT. Different views of the plot are shown below:



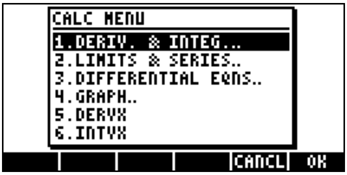


# Chapter 13
# Calculus Applications

In this Chapter we discuss applications of the calculator's functions to operations related to Calculus, e.g., limits, derivatives, integrals, power series, etc.

## The CALC (Calculus) menu

Many of the functions presented in this Chapter are contained in the calculator's CALC menu, available through the keystroke sequence ↰CALC (associated with the 4 key). The CALC menu shows the following entries:



The first four options in this menu are actually sub-menus that apply to (1) derivatives and integrals, (2) limits and power series, (3) differential equations, and (4) graphics. The functions in entries (1) and (2) will be presented in this Chapter. Differential equations, the subject of item (3), are presented in Chapter 16. Graphic functions, the subject of item (4), were presented at the end of Chapter 12. Finally, entries 5. DERVX and 6.INTVX are the functions to obtain a derivative and a indefinite integral for a function of the default CAS variable (typically, 'X'). Functions DERVX and INTVX are discussed in detail later.

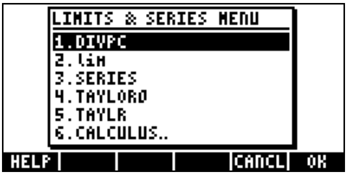
## Limits and derivatives

Differential calculus deals with derivatives, or rates of change, of functions and their applications in mathematical analysis. The derivative of a function is defined as a limit of the difference of a function as the increment in the independent variable tends to zero. Limits are used also to check the continuity of functions.

## Function lim

The calculator provides function *lim* to calculate limits of functions. This function uses as input an expression representing a function and the value where the limit is to be calculated. Function *lim* is available through the command catalog (↱ CAT ALPHA ↰ L) or through option 2. LIMITS & SERIES… of the CALC menu (see above).

**Note**: The functions available in the LIMITS & SERIES menu are shown next:

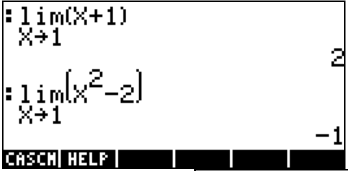


```
LIMITS & SERIES MENU
1.DIVPC
2.lim
3.SERIES
4.TAYLOR0
5.TAYLR
6.CALCULUS..
HELP |  |  |  | CANCL | OK
```

Function DIVPC is used to divide two polynomials producing a series expansion. Functions DIVPC, SERIES, TAYLOR0, and TAYLOR are used in series expansions of functions and discussed in more detail in this Chapter.

Function *lim* is entered in ALG mode as `lim(f(x),x=a)` to calculate the limit $\lim_{x \to a} f(x)$. In RPN mode, enter the function first, then the expression 'x=a', and finally function lim. Examples in ALG mode are shown next, including some limits to infinity. The keystrokes for the first example are as follows (using Algebraic mode, and system flag 117 set to CHOOSE boxes):

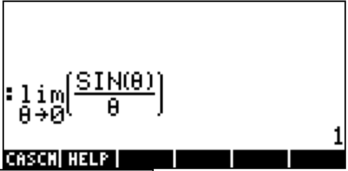
↰ CALC 2 OK 2 OK X + 1 ↱ , X ↱ = 1 ENTER



$:\lim_{X \to 1}(X+1)$ 2

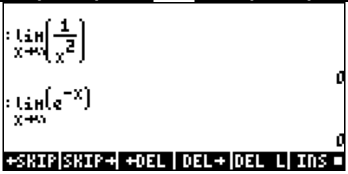
$:\lim_{X \to 1}(X^2-2)$ -1

CASCM | HELP



$:\lim_{\theta \to 0}\left(\frac{\text{SIN}(\theta)}{\theta}\right)$ 1

CASCM | HELP



$:\lim_{x \to \infty}\left(\frac{1}{x^2}\right)$ 0

$:\lim_{x \to \infty}\left(e^{-x}\right)$ 0

←SKIP | SKIP→ | ←DEL | DEL→ | DEL L | INS ■

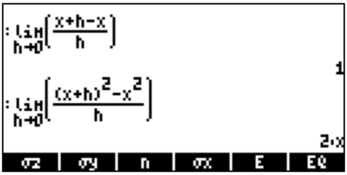
The infinity symbol is associated with the 0 key, i.e.., ↰ ∞ .

## Derivatives

The derivative of a function f(x) at x = a is defined as the limit

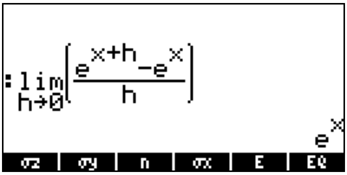
$$\frac{df}{dx} = f'(x) = \lim_{h->0} \frac{f(x+h)-f(x)}{h}$$

Some examples of derivatives using this limit are shown in the following screen shots:



$:\lim_{h\to 0}\left(\frac{x+h-x}{h}\right)$ $1$

$:\lim_{h\to 0}\left(\frac{(x+h)^2-x^2}{h}\right)$ $2\cdot x$

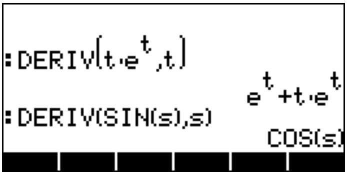


$:\lim_{h\to 0}\left(\frac{e^{x+h}-e^{x}}{h}\right)$ $e^{x}$

## Functions DERIV and DERVX

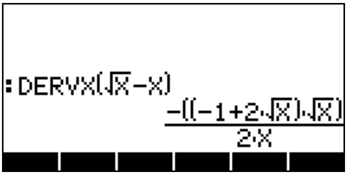
The function DERIV is used to take derivatives in terms of any independent variable, while the function DERVX takes derivatives with respect to the CAS default variable VX (typically 'X'). While function DERVX is available directly in the CALC menu, both functions are available in the DERIV.&INTEG sub-menu within the CALCL menu ( ↰ CALC ).

Function DERIV requires a function, say f(t), and an independent variable, say, t, while function DERVX requires only a function of VX. Examples are shown next in ALG mode. Recall that in RPN mode the arguments must be entered before the function is applied.



$:\mathrm{DERIV}(t\cdot e^{t},t)$ $e^{t}+t\cdot e^{t}$

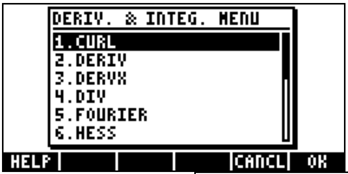
:DERIV(SIN(s),s) COS(s)

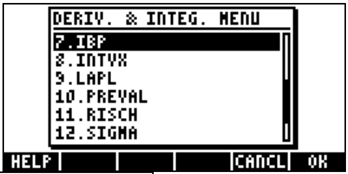


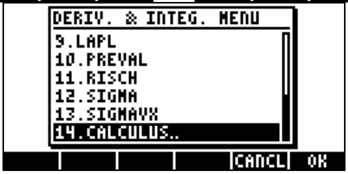
$:\mathrm{DERVX}(\sqrt{X}-X)$ $\frac{-((-1+2\cdot\sqrt{X})\cdot\sqrt{X})}{2\cdot X}$

## The DERIV&INTEG menu

The functions available in this sub-menu are listed below:





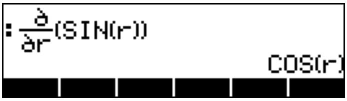


Out of these functions DERIV and DERVX are used for derivatives. The other functions include functions related to anti-derivatives and integrals (IBP, INTVX, PREVAL, RISCH, SIGMA, and SIGMAVX), to Fourier series (FOURIER),and to vector analysis (CURL, DIV, HESS, LAPL). Next we discuss functions DERIV and DERVX, the remaining functions are presented either later in this Chapter or in subsequent Chapters.

## Calculating derivatives with ∂

The symbol is available as ↱ ∂ (the COS key). This symbol can be used to enter a derivative in the stack or in the Equation Writer (see Chapter 2). If you use the symbol to write a derivative into the stack, follow it immediately with the independent variable, then by a pair of parentheses enclosing the function to be differentiated. Thus, to calculate the derivative d(sin(r),r), use, in ALG mode: ↱ ∂ ALPHA ↰ R ↰ () SIN ALPHA ↰ R ENTER

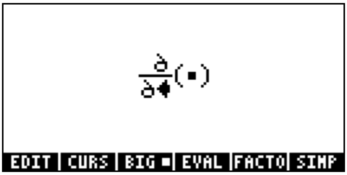
In RPN mode, this expression must be enclosed in quotes before entering it into the stack. The result in ALG mode is:



:$\frac{\partial}{\partial r}$(SIN(r))

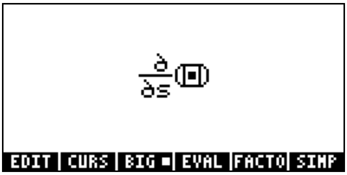
COS(r)

In the Equation Writer, when you press ↱ ∂ , the calculator provides the following expression:



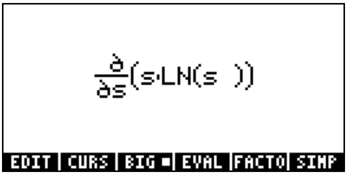
$$\frac{\partial}{\partial ◆}(■)$$

The insert cursor (◆) will be located right at the denominator awaiting for the user to enter an independent variable, say, s: ALPHA ↰ S . Then, press the right-arrow key (▶) to move to the placeholder between parentheses:



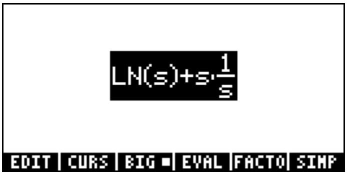
$$\frac{\partial}{\partial s}(■)$$

Next, enter the function to be differentiated, say, s*ln(s):



$$\frac{\partial}{\partial s}(s \cdot LN(s\ ))$$

To evaluate the derivative in the Equation Writer, press the up-arrow key ▲, four times, to select the entire expression, then, press EVAL. The derivative will be evaluated in the Equation Writer as:

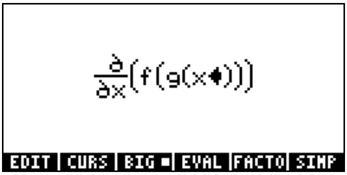


$$LN(s)+s \cdot \frac{1}{s}$$

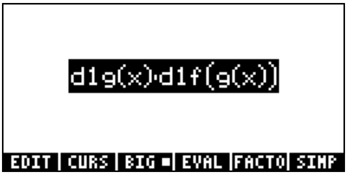
**Note**: The symbol $\partial$ is used formally in mathematics to indicate a partial derivative, i.e., the derivative of a function with more than one variable. However, the calculator does not distinguish between ordinary and partial derivatives, utilizing the same symbol for both. The user must keep this distinction in mind when translating results from the calculator to paper.

## The chain rule

The chain rule for derivatives applies to derivatives of composite functions. A general expression for the chain-rule is d{f[g(x)]}/dx = (df/dg)· (dg/dx). Using the calculator, this formula results in:

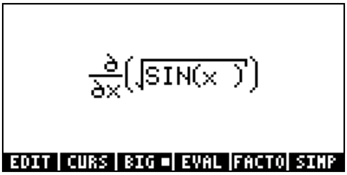


$\frac{\partial}{\partial x}(f(g(x)))$

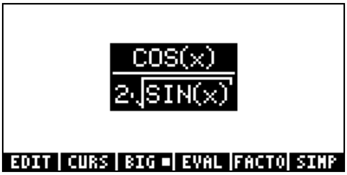


d1g(x)·d1f(g(x))

The terms d1 in front of g(x) and f(g(x)) in the expression above are abbreviations the calculator uses to indicate a first derivative when the independent variable, in this case x, is clearly defined. Thus, the latter result is interpreted as in the formula for the chain rule shown above. Here is another example of a chain rule application:



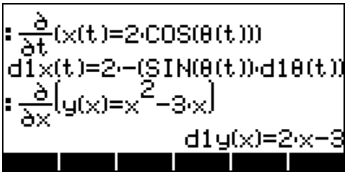
$\frac{\partial}{\partial x}(\sqrt{SIN(x)})$



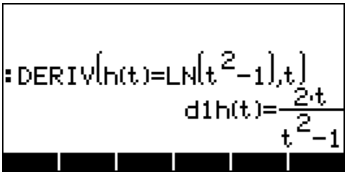
$\frac{COS(x)}{2\cdot\sqrt{SIN(x)}}$

## Derivatives of equations

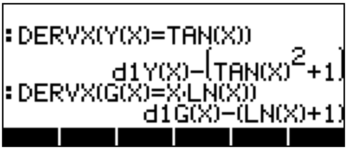
You can use the calculator to calculate derivatives of equations, i.e., expressions in which derivatives will exist in both sides of the equal sign. Some examples are shown below:



```
: ∂/∂t(x(t)=2·COS(θ(t)))
d1x(t)=2·-(SIN(θ(t))·d1θ(t))
: ∂/∂x(y(x)=x^2-3·x)
                 d1y(x)=2·x-3
```



```
:DERIV(h(t)=LN(t^2-1),t)
              d1h(t)=2·t/(t^2-1)
```

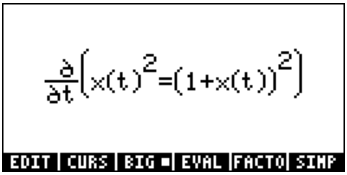


```
:DERVX(Y(X)=TAN(X))
         d1Y(X)-(TAN(X)^2+1)
:DERVX(G(X)=X·LN(X))
            d1G(X)-(LN(X)+1)
```

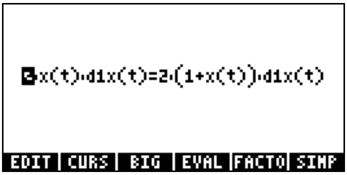
Notice that in the expressions where the derivative sign (∂) or function DERIV was used, the equal sign is preserved in the equation, but not in the cases where function DERVX was used. In these cases, the equation was re-written with all its terms moved to the left-hand side of the equal sign. Also, the equal sign was removed, but it is understood that the resulting expression is equal to zero.

## Implicit derivatives

Implicit derivatives are possible in expressions such as:



$$\frac{\partial}{\partial t}\left(x(t)^2=(1+x(t))^2\right)$$



```
2·x(t)·d1x(t)=2·(1+x(t))·d1x(t)
```

# Application of derivatives

Derivatives can be used for analyzing the graphs of functions and for optimizing functions of one variable (i.e., finding maxima and minima). Some applications of derivatives are shown next.

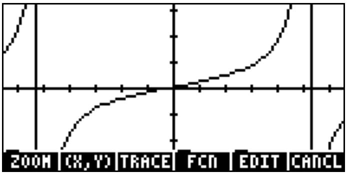
## Analyzing graphics of functions

In Chapter 11 we presented some functions that are available in the graphics screen for analyzing graphics of functions of the form y = f(x). These functions include (X,Y) and TRACE for determining points on the graph, as well as functions in the ZOOM and FCN menu. The functions in the ZOOM menu allow the user to zoom in into a graph to analyze it in more detail. These functions are described in detail in Chapter 12. Within the functions of the FCN menu, we can use the functions SLOPE, EXTR, F′, and TANL to determine the slope of a tangent to the graph, the extrema (minima and maxima) of the function, to plot the derivative, and to find the equation of the tangent line.

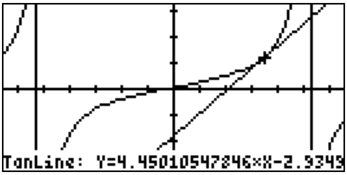
Try the following example for the function y = tan(x).

- Press ↰ 2D/3D, simultaneously in RPN mode, to access to the PLOT SETUP window.
- Change `TYPE` to `FUNCTION`, if needed, by using [CHOOS].
- Press ▼ and type in the equation 'TAN(X)'.
- Make sure the independent variable is set to 'X'.
- Press NXT OK to return to normal calculator display.
- Press ↰ WIN, simultaneously, to access the PLOT window
- Change H-VIEW range to –2 to 2, and V-VIEW range to –5 to 5.
- Press ERASE DRAW to plot the function in polar coordinates.

The resulting plot looks as follows:



- Notice that there are vertical lines that represent asymptotes. These are not part of the graph, but show points where TAN(X) goes to ± ∞ at certain values of X.
- Press TRACE (X,Y), and move the cursor to the point X: 1.08E0, Y: 1.86E0. Next, press NXT FCN SLOPE. The result is Slope: 4.45010547846.
- Press NXT NXT TANL. This operation produces the equation of the tangent line, and plots its graph in the same figure. The result is shown in the figure below:

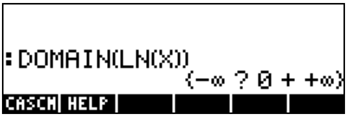


TanLine: Y=4.45010547846×X-2.9343

- Press NXT PICT CANCL ON to return to normal calculator display. Notice that the slope and tangent line that you requested are listed in the stack.

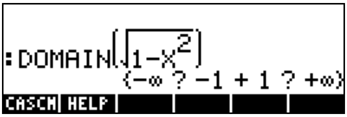
## Function DOMAIN

Function DOMAIN, available through the command catalog (↱ _CAT_ ), provides the domain of definition of a function as a list of numbers and specifications. For example,



```
:DOMAIN(LN(X))
                {-∞ ? 0 + +∞}
```

indicates that between –∞ and 0, the function LN(X) is not defined (?), while from 0 to +∞, the function is defined (+). On the other hand,



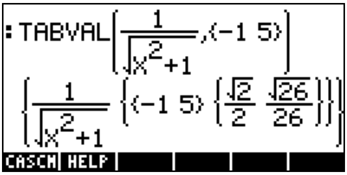
$$:\mathrm{DOMAIN}\left(\sqrt{1-X^2}\right)$$

$$\{-\infty\ ?\ -1\ +\ 1\ ?\ +\infty\}$$

indicates that the function is not defined between –∞ and -1, nor between 1 and +∞. The domain of this function is, therefore, -1<X<1.

## Function TABVAL

This function is accessed through the command catalog or through the GRAPH sub-menu in the CALC menu. Function TABVAL takes as arguments a function of the CAS variable, f(X), and a list of two numbers representing a domain of interest for the function f(X). Function TABVAL returns the input values plus the range of the function corresponding to the domain used as input. For example,



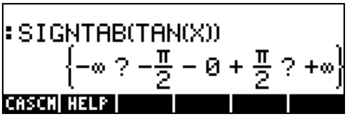
$$:\mathrm{TABVAL}\left(\frac{1}{\sqrt{X^2+1}},\{-1\ 5\}\right)$$

$$\left\{\frac{1}{\sqrt{X^2+1}}\ \left\{\{-1\ 5\}\ \left\{\frac{\sqrt{2}}{2}\ \frac{\sqrt{26}}{26}\right\}\right\}\right\}$$

This result indicates that the range of the function $f(X) = \frac{1}{\sqrt{X^2+1}}$ corresponding to the domain D = { -1,5 } is R = $\left\{\frac{\sqrt{2}}{2}, \frac{\sqrt{26}}{26}\right\}$.

## Function SIGNTAB

Function SIGNTAB, available through the command catalog (↱ _CAT_ ), provides information on the sign of a function through its domain. For example, for the TAN(X) function,

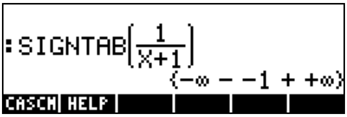


```
:SIGNTAB(TAN(X))
```

$\left\{-\infty\ ?\ -\frac{\pi}{2}\ -\ 0\ +\ \frac{\pi}{2}\ ?\ +\infty\right\}$

SIGNTAB indicates that TAN(X) is negative between –π/2 and 0, and positive between 0 and π /2. For this case, SIGNTAB does not provide information (?) in the intervals between –∞ and -π /2, nor between +π /2 and ∞. Thus, SIGNTAB, for this case, provides information only on the main domain of TAN(X), namely -π /2 < X < +π /2.

A second example of function SIGNTAB is shown below:



:SIGNTAB$\left(\frac{1}{X+1}\right)$

```
{-∞ - -1 + +∞}
```

For this case, the function is negative for X<-1 and positive for X> -1.

## Function TABVAR

This function is accessed through the command catalog or through the GRAPH sub-menu in the CALC menu. It uses as input the function f(VX), where VX is the default CAS variable. The function returns the following, in RPN mode:

- Level 3: the function f(VX)

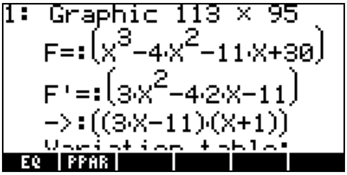
- Two lists, the first one indicates the variation of the function (i.e., where it increases or decreases) in terms of the independent variable VX, the second one indicates the variation of the function in terms of the dependent variable.

- A graphic object showing how the variation table was computed.

Example: Analyze the function Y = $X^3$-4$X^2$-11X+30, using the function TABVAR. Use the following keystrokes, in RPN mode:

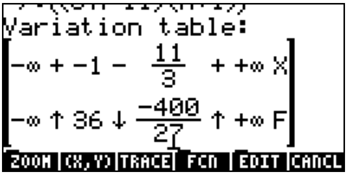
```
'X^3-4*X^2-11*X+30' ENTER ↱ CAT ALPHA T (select TABVAR) OK
```

This is what the calculator shows in stack level 1:



```
1: Graphic 113 × 95
   F=:(X^3-4·X^2-11·X+30)
   F'=:(3·X^2-4·2·X-11)
   ->:((3·X-11)·(X+1))
   Variation table:
```

This is a graphic object. To be able to the result in its entirety, press ▼. The variation table of the function is shown as follows:



Variation table:

$$\begin{bmatrix} -\infty & + & -1 & - & \frac{11}{3} & + & +\infty & X \\ -\infty & \uparrow & 36 & \downarrow & \frac{-400}{27} & \uparrow & +\infty & F \end{bmatrix}$$

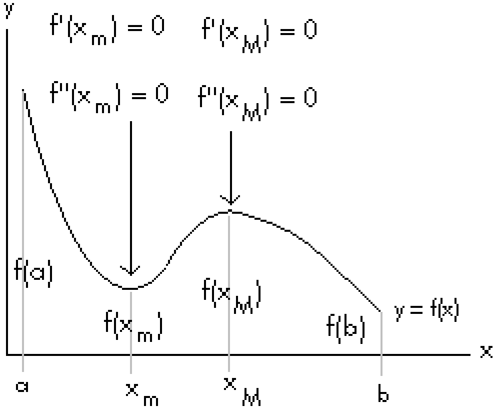
Press ON to recover normal calculator display. Press ◄ to drop this last result from the stack.

Two lists, corresponding to the top and bottom rows of the graphics matrix shown earlier, now occupy level 1. These lists may be useful for programming purposes. Press ◄ to drop this last result from the stack.

The interpretation of the variation table shown above is as follows: the function F(X) increases for X in the interval (-∞, -1), reaching a maximum equal to 36 at X = -1. Then, F(X) decreases until X = 11/3, reaching a minimum of –400/27. After that F(X) increases until reaching +∞. Also, at X = ±∞, F(X)= ±∞.

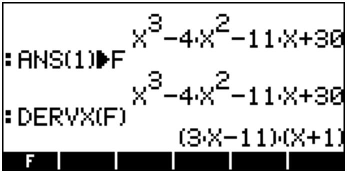
## Using derivatives to calculate extreme points

"Extreme points," or extrema, is the general designation for maximum and minimum values of a function in a given interval. Since the derivative of a function at a given point represents the slope of a line tangent to the curve at that point, then values of x for which f'(x) =0 represent points where the graph of the function reaches a maximum or minimum. Furthermore, the value of the second derivative of the function, f"(x), at those points determines whether the point is a *relative or local maximum* [f"(x)<0] or *minimum* [f"(x)>0]. These ideas are illustrated in the figure below.

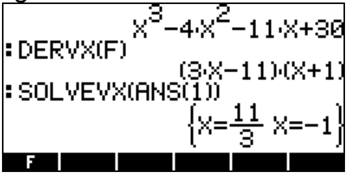


In this figure we limit ourselves to determining extreme points of the function y = f(x) in the x-interval [a,b]. Within this interval we find two points, x = $x_m$ and x = $x_M$, where f'(x)=0. The point x = $x_m$, where f"(x)>0, represents a local minimum, while the point x = $x_M$, where f"(x)<0, represents a local maximum. From the graph of y = f(x) it follows that the absolute maximum in the interval [a,b] occurs at x = a, while the absolute minimum occurs at x = b.

For example, to determine where the critical points of function 'X^3-4*X^2-11*X+30' occur, we can use the following entries in ALG mode:

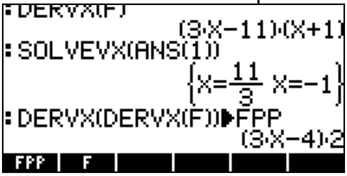


```
              X^3-4·X^2-11·X+30
:ANS(1)▶F
              X^3-4·X^2-11·X+30
:DERVX(F)
                 (3·X-11)·(X+1)
 F
```

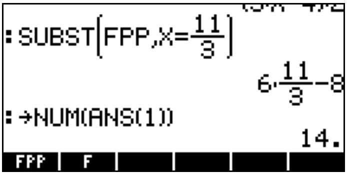


```
              X^3-4·X^2-11·X+30
:DERVX(F)
                 (3·X-11)·(X+1)
:SOLVEVX(ANS(1))
                  {X=11/3 X=-1}
 F
```

We find two critical points, one at x = 11/3 and one at x = -1. To evaluate the second derivative at each point use:

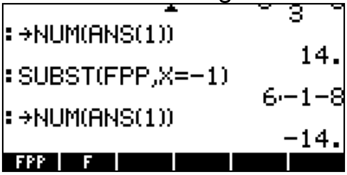


```
:DERVX(F)
                 (3·X-11)·(X+1)
:SOLVEVX(ANS(1))
                  {X=11/3 X=-1}
:DERVX(DERVX(F))▶FPP
                      (3·X-4)·2
 FPP | F
```



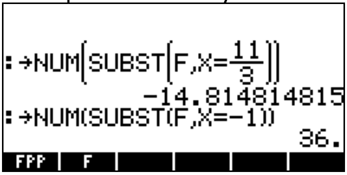
```
:SUBST(FPP,X=11/3)
                       6·11/3-8
:→NUM(ANS(1))
                            14.
 FPP | F
```

The last screen shows that f"(11/3) = 14, thus, x = 11/3 is a relative minimum. For x = -1, we have the following:



```
:→NUM(ANS(1))
                            14.
:SUBST(FPP,X=-1)
                         6·-1-8
:→NUM(ANS(1))
                           -14.
 FPP | F
```

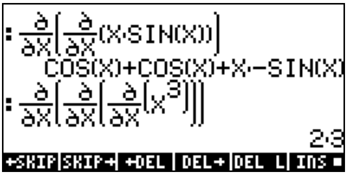
This result indicates that f"(-1) = -14, thus, x = -1 is a relative maximum. Evaluate the function at those points to verify that indeed f(-1) > f(11/3).



```
:→NUM(SUBST(F,X=11/3))
                  -14.814814815
:→NUM(SUBST(F,X=-1))
                            36.
 FPP | F
```

## Higher order derivatives

Higher order derivatives can be calculated by applying a derivative function several times, e.g.,



$: \frac{\partial}{\partial X}\left(\frac{\partial}{\partial X}(X\cdot SIN(X))\right)$

$COS(X)+COS(X)+X\cdot -SIN(X)$

$: \frac{\partial}{\partial X}\left(\frac{\partial}{\partial X}\left(\frac{\partial}{\partial X}\left(X^3\right)\right)\right)$

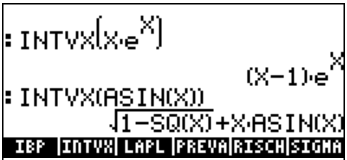
$2\cdot 3$

## Anti-derivatives and integrals

An anti-derivative of a function f(x) is a function F(x) such that f(x) = dF/dx. For example, since d($x^3$) /dx = $3x^2$, an anti-derivative of f(x) = $3x^2$ is F(x) = $x^3$ + C, where C is a constant. One way to represent an anti-derivative is as a indefinite integral, i.e., $\int f(x)dx = F(x) + C$, if and only if, f(x) = dF/dx, and C = constant.

### Functions INT, INTVX, RISCH, SIGMA and SIGMAVX

The calculator provides functions INT, INTVX, RISCH, SIGMA and SIGMAVX to calculate anti-derivatives of functions. Functions INT, RISCH, and SIGMA work with functions of any variable, while functions INTVX, and SIGMAVX utilize functions of the CAS variable VX (typically, 'x'). Functions INT and RISCH require, therefore, not only the expression for the function being integrated, but also the independent variable name. Function INT, requires also a value of x where the anti-derivative will be evaluated. Functions INTVX and SIGMAVX require only the expression of the function to integrate in terms of VX. Some examples are shown next in ALG mode:

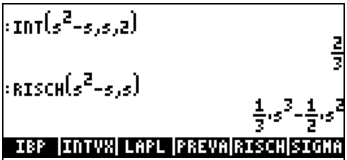


$: INTVX\left(X\cdot e^X\right)$

$(X-1)\cdot e^X$

$: INTVX(ASIN(X))$

$\sqrt{1-SQ(X)}+X\cdot ASIN(X)$

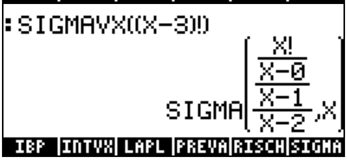


$: INT\left(s^2-s,s,2\right)$

$\frac{2}{3}$

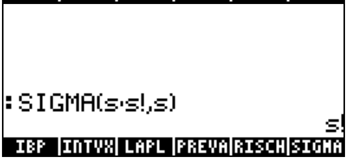
$: RISCH\left(s^2-s,s\right)$

$\frac{1}{3}\cdot s^3-\frac{1}{2}\cdot s^2$



$: SIGMAVX((X-3)!)$

$SIGMA\left(\frac{\frac{X!}{X-0}}{\frac{X-1}{X-2}},X\right)$



$: SIGMA(s\cdot s!,s)$

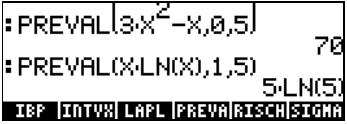
$s!$

Please notice that functions SIGMAVX and SIGMA are designed for integrands that involve some sort of integer function like the factorial (!) function shown above. Their result is the so-called discrete derivative, i.e., one defined for integer numbers only.

## Definite integrals

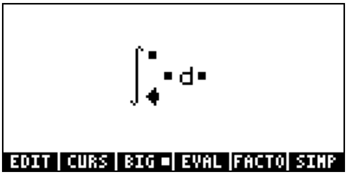
In a definite integral of a function, the resulting anti-derivative is evaluated at the upper and lower limit of an interval (a,b) and the evaluated values subtracted. Symbolically, $\int_a^b f(x)dx = F(b) - F(a),$ with f(x) = dF/dx.

To calculate definite integrals of functions using the CAS variable VX (typically, 'X'), use function PREVAL(f(x),a,b). For example,

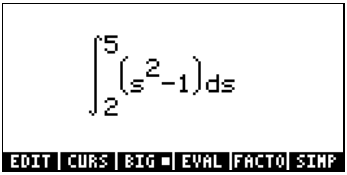


```
:PREVAL(3·X^2-X,0,5)
                   70
:PREVAL(X·LN(X),1,5)
              5·LN(5)
```

To calculate definite integrals the calculator also provides the integral symbol as the keystroke combination ↱ ∫ (associated with the TAN key). The simplest way to build an integral is by using the Equation Writer (see Chapter 2 for an example). Within the Equation Writer, the symbol ↱ ∫ produces the integral sign and provides placeholders for the integration limits (a,b), for the function, f(x), and for the variable of integration (x). The following screen shots show how to build a particular integral. The insert cursor is first located in the lower limit of integration, enter a value and press the right-arrow key (▶) to move to the upper limit of integration. Enter a value in that location and press ▶ again to move to the integrand location. Type the integrand expression, and press once more to move to the differential place holder, type the variable of integration in that location and the integral is ready to be calculated.

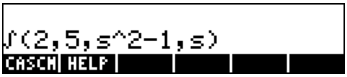


$$\int_{◆}^{■} ■ \, d■$$



$$\int_2^5 (s^2-1)ds$$

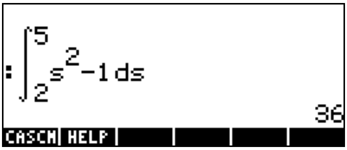
At this point, you can press ENTER to return the integral to the stack, which will show the following entry (ALG mode shown):



```
∫(2,5,s^2-1,s)
```

This is the general format for the definite integral when typed directly into the stack, i.e., ∫(lower limit, upper limit, integrand, variable of integration)

Pressing ENTER at this point will evaluate the integral in the stack:



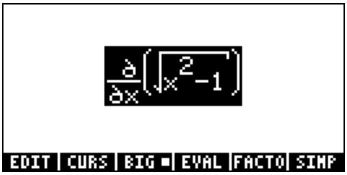
$$:\int_{2}^{5} s^{2}-1\,ds$$

36

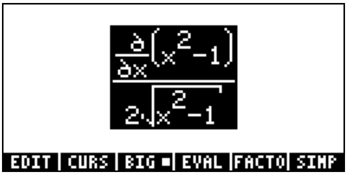
The integral can be evaluated also in the Equation Writer by selecting the entire expression an using the soft menu key EVAL.

## Step-by-step evaluation of derivatives and integrals

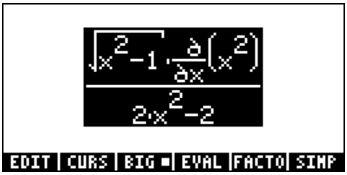
With the Step/Step option in the CAS MODES windows selected (see Chapter 1), the evaluation of derivatives and integrals will be shown step by step. For example, here is the evaluation of a derivative in the Equation Writer:



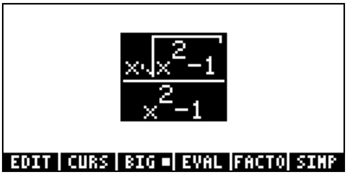
$$\frac{\partial}{\partial x}\left(\sqrt{x^{2}-1}\right)$$



$$\frac{\frac{\partial}{\partial x}\left(x^{2}-1\right)}{2\cdot\sqrt{x^{2}-1}}$$



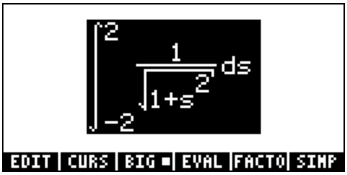
$$\frac{\sqrt{x^{2}-1}\cdot\frac{\partial}{\partial x}\left(x^{2}\right)}{2\cdot x^{2}-2}$$



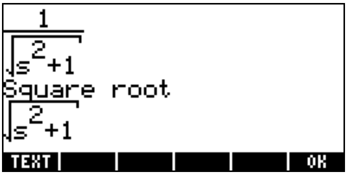
$$\frac{x\cdot\sqrt{x^{2}-1}}{x^{2}-1}$$

Notice the application of the chain rule in the first step, leaving the derivative of the function under the integral explicitly in the numerator. In the second step, the resulting fraction is rationalized (eliminating the square root from the denominator), and simplified. The final version is shown in the third step. Each step is shown by pressing the EVAL menu key, until reaching the point where further application of function EVAL produce no more changes in the expression.

The following example shows the evaluation of a definite integral in the Equation Writer, step-by-step:



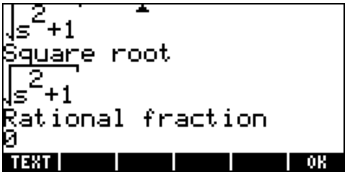
$$\int_{-2}^{2}\frac{1}{\sqrt{1+s^2}}ds$$



$\frac{1}{\sqrt{s^2+1}}$

Square root

$\sqrt{s^2+1}$



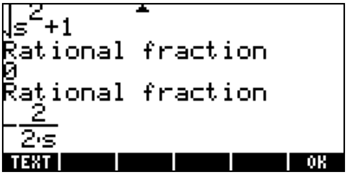
$\sqrt{s^2+1}$

Square root

$\sqrt{s^2+1}$

Rational fraction

0



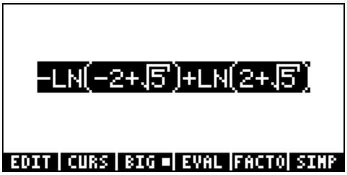
$\sqrt{s^2+1}$

Rational fraction

0

Rational fraction

$\frac{2}{2\cdot s}$

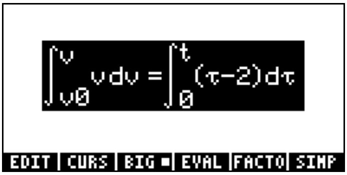


$-\text{LN}(-2+\sqrt{5})+\text{LN}(2+\sqrt{5})$

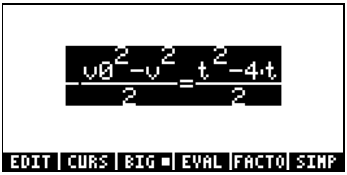
Notice that the step-by-step process provides information on the  intermediate steps followed by the CAS to solve this integral.  First, CAS identifies a square root integral, next, a rational fraction, and a second rational expression, to come up with the final result.   Notice that these steps make a lot of sense to the calculator, although not enough information is provided to the user on the individual steps.

## Integrating an equation

Integrating an equation is straightforward, the calculator simply integrates both sides of the equation simultaneously, e.g.,



$\int_{v0}^{v} v\,dv = \int_{0}^{t} (\tau-2)\,d\tau$



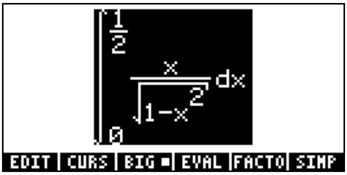
$\frac{v0^2-v^2}{2}=\frac{t^2-4\cdot t}{2}$

## Techniques of integration

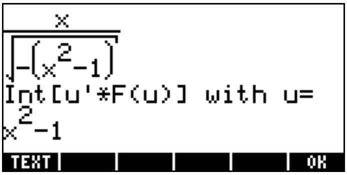
Several techniques of integration can be implemented in the calculators, as shown in the following examples.

### Substitution or change of variables

Suppose we want to calculate the integral $\int_0^2 \frac{x}{\sqrt{1-x^2}}dx$. If we use step-by-step calculation in the Equation Writer, this is the sequence of variable substitutions:



$\int_0^{\frac{1}{2}} \frac{x}{\sqrt{1-x^2}}dx$

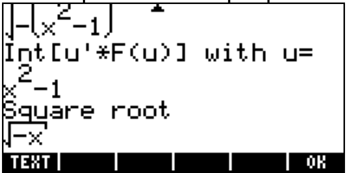


$\frac{x}{\sqrt{-(x^2-1)}}$

Int[u'*F(u)] with u=

$x^2-1$

This second step shows the proper substitution to use, u = $x^2$-1.



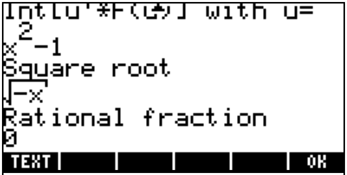
$\sqrt{-(x^2-1)}$

Int[u'*F(u)] with u=

$x^2-1$

Square root

$\sqrt{-x}$



Int[u'*F(u)] with u=

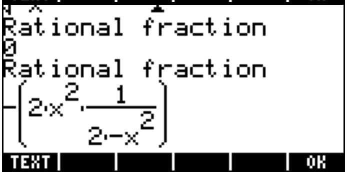
$x^2-1$

Square root

$\sqrt{-x}$

Rational fraction

0

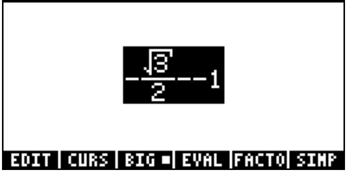


Rational fraction

0

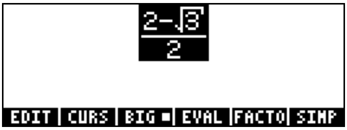
Rational fraction

$-\left(2\cdot x^2\cdot\frac{1}{2\cdot -x^2}\right)$



$-\frac{\sqrt{3}}{2}--1$

The last four steps show the progression of the solution: a square root, followed by a fraction, a second fraction, and the final result. This result can be simplified by using function EVAL, to read:



$\frac{2-\sqrt{3}}{2}$

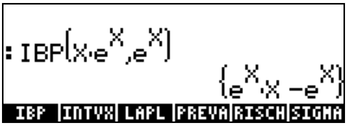
**Integration by parts and differentials**

A differential of a function y = f(x), is defined as dy = f'(x) dx, where f'(x) is the derivative of f(x). Differentials are used to represent small increments in the variables. The differential of a product of two functions, y = u(x)v(x), is given by dy = u(x)dv(x) +du(x)v(x), or, simply, d(uv) = udv - vdu. Thus, the integral of udv = d(uv) - vdu, is written as $\int udv = \int d(uv) - \int vdu$ . Since by the definition of a differential, ∫dy = y, we write the previous expression as

$$\int udv = uv - \int vdu \; .$$

This formulation, known as integration by parts, can be used to find an integral if dv is easily integrable. For example, the integral $\int xe^x dx$ can be solved by integration by parts if we use u = x, dv = $e^x$dx, since, v = $e^x$. With du = dx, the integral becomes $\int xe^x dx = \int udv = uv - \int vdu = xe^x - \int e^x dx = xe^x - e^x$.

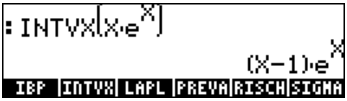
The calculator provides function IBP, under the CALC/DERIV&INTG menu, that takes as arguments the original function to integrate, namely, u(X)*v'(X), and the function v(X), and returns u(X)*v(X) and -v(X)*u'(X). In other words, function IBP returns the two terms of the right-hand side in the integration by parts equation. For the example used above, we can write in ALG mode:



```
:IBP(X·e^X,e^X)
                    {e^X·X  -e^X}
```

Thus, we can use function IBP to provide the components of an integration by parts. The next step will have to be carried out separately.

It is important to mention that the integral can be calculated directly by using, for example,



$$:\mathrm{INTVX}(X\cdot e^{X})$$

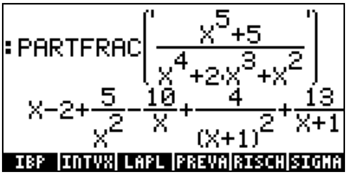
$$(X-1)\cdot e^{X}$$

**Integration by partial fractions**

Function PARTFRAC, presented in Chapter 5, provides the decomposition of a fraction into partial fractions. This technique is useful to reduce a complicated fraction into a sum of simple fractions that can then be integrated term by term. For example, to integrate

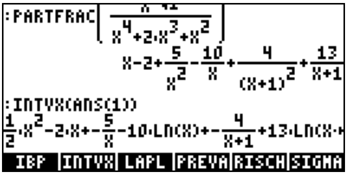
$$\int \frac{X^5+5}{X^4+2X^3+X}\,dX$$

we can decompose the fraction into its partial component fractions, as follows:



$$:\mathrm{PARTFRAC}\left(\frac{X^5+5}{X^4+2\cdot X^3+X^2}\right)$$

$$X-2+\frac{5}{X^2}-\frac{10}{X}+\frac{4}{(X+1)^2}+\frac{13}{X+1}$$



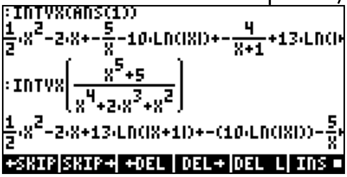
$$:\mathrm{PARTFRAC}\left(\frac{X^5+5}{X^4+2\cdot X^3+X^2}\right)$$

$$X-2+\frac{5}{X^2}-\frac{10}{X}+\frac{4}{(X+1)^2}+\frac{13}{X+1}$$

$$:\mathrm{INTVX}(\mathrm{ANS}(1))$$

$$\frac{1}{2}\cdot X^2-2\cdot X+-\frac{5}{X}-10\cdot \mathrm{LN}(X)+-\frac{4}{X+1}+13\cdot \mathrm{LN}(X\cdot$$

The direct integration produces the same result, with some switching of the terms (Rigorous mode set in the CAS – see Chapter 2):



$$:\mathrm{INTVX}(\mathrm{ANS}(1))$$

$$\frac{1}{2}\cdot X^2-2\cdot X+-\frac{5}{X}-10\cdot \mathrm{LN}(|X|)+-\frac{4}{X+1}+13\cdot \mathrm{LN}(|$$

$$:\mathrm{INTVX}\left(\frac{X^5+5}{X^4+2\cdot X^3+X^2}\right)$$

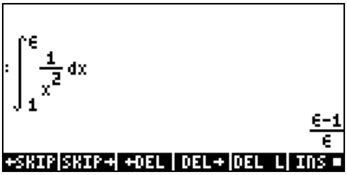
$$\frac{1}{2}\cdot X^2-2\cdot X+13\cdot \mathrm{LN}(|X+1|)+-(10\cdot \mathrm{LN}(|X|))-\frac{5}{X}$$

### Improper integrals

These are integrals with infinite limits of integration. Typically, an improper integral is dealt with by first calculating the integral as a limit to infinity, e.g.,

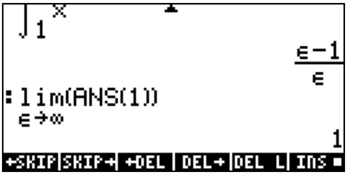
$$\int_1^\infty \frac{dx}{x^2} = \lim_{\varepsilon\to\infty}\int_1^\varepsilon \frac{dx}{x^2}.$$

Using the calculator, we proceed as follows:



$$:\int_1^{\epsilon}\frac{1}{x^2}\,dx$$

$$\frac{\epsilon-1}{\epsilon}$$

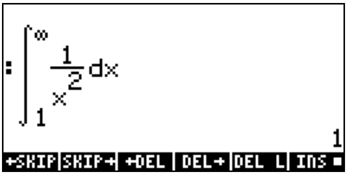


$$\frac{\epsilon-1}{\epsilon}$$

$$:\lim_{\epsilon\to\infty}(\mathrm{ANS}(1))$$

1

Alternatively, you can evaluate the integral to infinity from the start, e.g.,

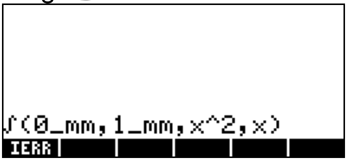


$$:\int_1^{\infty}\frac{1}{x^2}\,dx$$

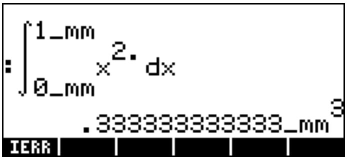
1

## Integration with units

An integral can be calculated with units incorporated into the limits of integration, as in the example shown below that uses ALG mode, with the CAS set to Approx mode. The left-hand side figure shows the integral typed in the line editor before pressing ENTER. The right-hand figure shows the result after pressing ENTER.



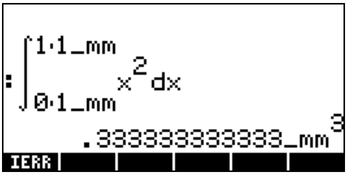
```
∫(0_mm,1_mm,x^2,x)
```



$$:\int_{0\_mm}^{1\_mm} x^{2.}\,dx$$

$$.333333333333\_mm^3$$

If you enter the integral with the CAS set to Exact mode, you will be asked to change to Approx mode, however, the limits of the integral will be shown in a different format as shown here:

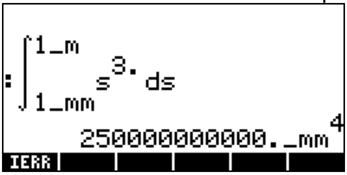


$$:\int_{0\cdot 1\_mm}^{1\cdot 1\_mm} x^2 dx$$

.333333333333_mm$^3$

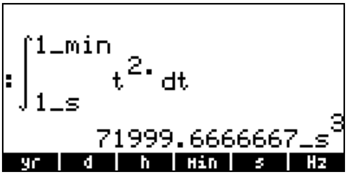
These limits represent 1×1_mm and 0×1_mm, which is the same as 1_mm and 0_mm, as before. Just be aware of the different formats in the output.

Some notes in the use of units in the limits of integrations:
1 – The units of the lower limit of integration will be the ones used in the final result, as illustrated in the two examples below:



$$:\int_{1\_mm}^{1\_m} s^{3.} ds$$

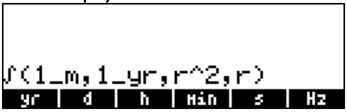
250000000000._mm$^4$



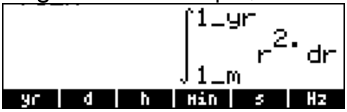
$$:\int_{1\_s}^{1\_min} t^{2.} dt$$

71999.6666667_s$^3$

2 - Upper limit units must be consistent with lower limit units. Otherwise, the calculator simply returns the unevaluated integral. For example,

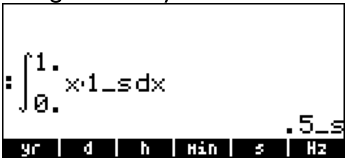


∫(1_m,1_yr,r^2,r)



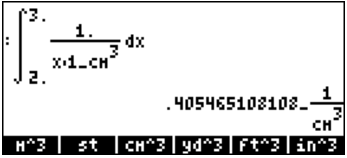
$$\int_{1\_m}^{1\_yr} r^{2.} dr$$

3 – The integrand may have units too. For example:



$$:\int_{0.}^{1.} x\cdot 1\_s dx$$

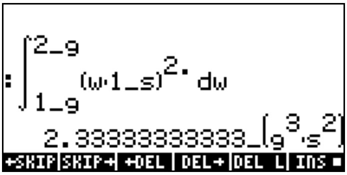
.5_s



$$:\int_{2.}^{3.} \frac{1.}{x\cdot 1\_cm^3} dx$$

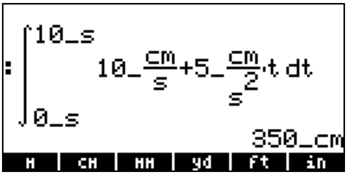
.405465108108_$\frac{1}{cm^3}$

4 – If both the limits of integration and the integrand have units, the resulting units are combined according to the rules of integration. For example,



$$:\int_{1\_g}^{2\_g} (w \cdot 1\_s)^{2.}\,dw$$

$$2.33333333333\_\left(g^3 \cdot s^2\right)$$



$$:\int_{0\_s}^{10\_s} 10\_\frac{cm}{s}+5\_\frac{cm}{s^2}\cdot t\,dt$$

$$350\_cm$$

# Infinite series

An infinite series has the form $\sum_{n=0,1}^{\infty} h(n)(x-a)^n$. The infinite series typically starts with indices n = 0 or n = 1. Each term in the series has a coefficient h(n) that depends on the index n.

## Taylor and Maclaurin's series

A function f(x) can be expanded into an infinite series around a point x=x$_0$ by using a Taylor's series, namely,

$$f(x)=\sum_{n=0}^{\infty}\frac{f^{(n)}(x_o)}{n!}\cdot(x-x_o)^n\ ,$$

where f$^{(n)}$(x) represents the n-th derivative of f(x) with respect to x, f$^{(0)}$(x) = f(x).

If the value x$_0$ is zero, the series is referred to as a Maclaurin's series, i.e.,

$$f(x)=\sum_{n=0}^{\infty}\frac{f^{(n)}(0)}{n!}\cdot x^n$$

## Taylor polynomial and reminder

In practice, we cannot evaluate all terms in an infinite series, instead, we approximate the series by a polynomial of order k, P$_k$(x), and estimate the order of a residual, R$_k$(x), such that

$$f(x)=\sum_{n=0}^{k}\frac{f^{(n)}(x_o)}{n!}\cdot(x-x_o)^n+\sum_{n=k+1}^{\infty}\frac{f^{(n)}(x_o)}{n!}\cdot(x-x_o)^n,$$

i.e., $$f(x)=P_k(x)+R_k(x).$$

The polynomial $P_k(x)$ is referred to as Taylor's polynomial. The order of the residual is estimated in terms of a small quantity $h = x-x_0$, i.e., evaluating the polynomial at a value of x very close to $x_0$. The residual if given by

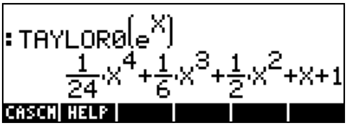
$$R_k(x)=\frac{f^{(k+1)}(\xi)}{k!}\cdot h^{k+1},$$

where $\xi$ is a number near $x = x_0$. Since $\xi$ is typically unknown, instead of an estimate of the residual, we provide an estimate of the order of the residual in reference to h, i.e., we say that $R_k(x)$ has an error of order $h^{n+1}$, or $R \approx O(h^{k+1})$. If h is a small number, say, h<<1, then $h^{k+1}$ will be typically very small, i.e., $h^{k+1}<<h^k<< \ldots << h << 1$. Thus, for x close to $x_0$, the larger the number of elements in the Taylor polynomial, the smaller the order of the residual.

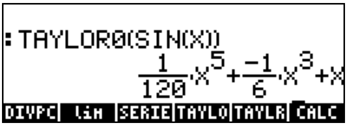
## Functions TAYLR, TAYLR0, and SERIES

Functions TAYLR, TAYLR0, and SERIES are used to generate Taylor polynomials, as well as Taylor series with residuals. These functions are available in the CALC/LIMITS&SERIES menu described earlier in this Chapter.

Function TAYLOR0 performs a Maclaurin series expansion, i.e., about X = 0, of an expression in the default independent variable, VX (typically 'X'). The expansion uses a 4-th order relative power, i.e., the difference between the highest and lowest power in the expansion is 4. For example,

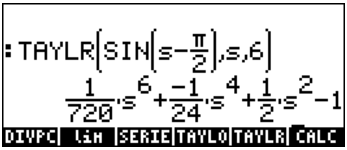


```
:TAYLOR0(e^X)
```
$$\frac{1}{24}\cdot X^4+\frac{1}{6}\cdot X^3+\frac{1}{2}\cdot X^2+X+1$$



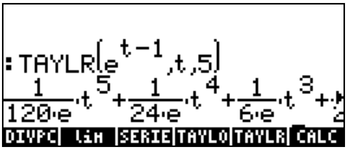
```
:TAYLOR0(SIN(X))
```
$$\frac{1}{120}\cdot X^5+\frac{-1}{6}\cdot X^3+X$$

Function TAYLR produces a Taylor series expansion of a function of any variable x about a point x = a for the order k specified by the user. Thus, the function has the format TAYLR(f(x-a),x,k). For example,



:TAYLR$\left(\mathrm{SIN}\left(s-\frac{\pi}{2}\right),s,6\right)$

$\frac{1}{720}\cdot s^6+\frac{-1}{24}\cdot s^4+\frac{1}{2}\cdot s^2-1$



:TAYLR$\left(e^{t-1},t,5\right)$

$\frac{1}{120\cdot e}\cdot t^5+\frac{1}{24\cdot e}\cdot t^4+\frac{1}{6\cdot e}\cdot t^3+$

Function SERIES produces a Taylor polynomial using as arguments the function f(x) to be expanded, a variable name alone (for Maclaurin's series) or an expression of the form 'variable = value' indicating the point of expansion of a Taylor series, and the order of the series to be produced. Function SERIES returns two output items a list with four items, and an expression for h = x - a, if the second argument in the function call is 'x=a', i.e., an expression for the increment h. The list returned as the first output object includes the following items:

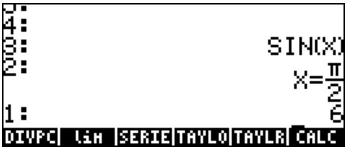
1 - Bi-directional limit of the function at point of expansion, i.e., $\lim_{x \to a} f(x)$

2 - An equivalent value of the function near x = a

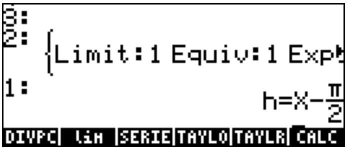
3 - Expression for the Taylor polynomial

4 - Order of the residual or remainder

Because of the relatively large amount of output, this function is easier to handle in RPN mode. For example:

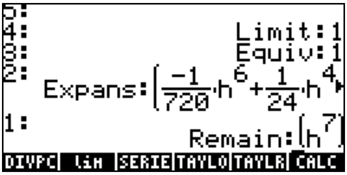


```
4:
3:                SIN(X)
2:                 X=π/2
1:                     6
```

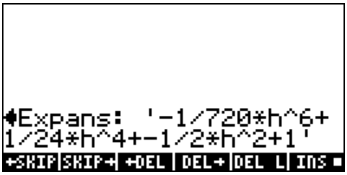


```
3:
2:  {Limit:1 Equiv:1 Exp
1:
              h=X-π/2
```

Drop the contents of stack level 1 by pressing ◄, and then enter EVAL, to decompose the list. The results are as follows:



```
5:
4:                 Limit:1
3:                 Equiv:1
2:  Expans:(-1/720·h^6+1/24·h^4▸
1:
               Remain:(h^7)
```



```
◂Expans: '-1/720*h^6+
1/24*h^4+-1/2*h^2+1'
```

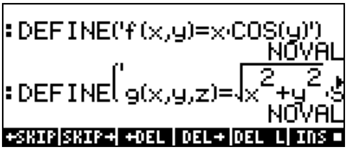
In the right-hand side figure above, we are using the line editor to see the series expansion in detail.

# Chapter 14
# Multi-variate Calculus Applications

Multi-variate calculus refers to functions of two or more variables. In this Chapter we discuss the basic concepts of multi-variate calculus including partial derivatives and multiple integrals.

## Multi-variate functions

A function of two or more variables can be defined in the calculator by using the DEFINE function (← DEF ). To illustrate the concept of partial derivative, we will define a couple of multi-variate functions, f(x,y) = x cos(y), and g(x,y,z) = $(x^2+y^2)^{1/2}$sin(z), as follows:



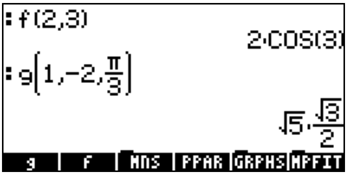
:DEFINE('f(x,y)=x·COS(y)')

NOVAL

:DEFINE$\left('g(x,y,z)=\sqrt{x^2+y^2}\cdot S\right.$

NOVAL

We can evaluate the functions as we would evaluate any other calculator function, e.g.,



:f(2,3)

2·COS(3)

:$g\left(1,-2,\frac{\pi}{3}\right)$

$\sqrt{5}\cdot\frac{\sqrt{3}}{2}$

Graphics of two-dimensional functions are possible using Fast3D, Wireframe, Ps-Contour, Y-Slice, Gridmap, and Pr-Surface plots as described in Chapter 12.

## Partial derivatives

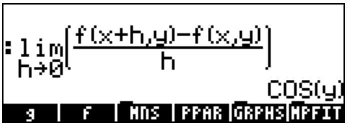
Consider the function of two variables z = f(x,y), the partial derivative of the function with respect to x is defined by the limit

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h,y) - f(x,y)}{h} \quad .$$

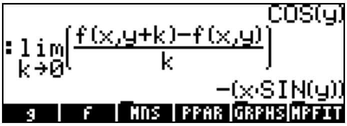
Similarly,

$$\frac{\partial f}{\partial y} = \lim_{k \to 0} \frac{f(x,y+k) - f(x,y)}{k} .$$

We will use the multi-variate functions defined earlier to calculate partial derivatives using these definitions. Here are the derivatives of f(x,y) with respect to x and y, respectively:



```
:lim  [f(x+h,y)-f(x,y)]
 h→0  [       h        ]
                  COS(y)
```



```
                  COS(y)
:lim  [f(x,y+k)-f(x,y)]
 k→0  [       k        ]
             -(x·SIN(y))
```

Notice that the definition of partial derivative with respect to x, for example, requires that we keep y fixed while taking the limit as h→0. This suggest a way to quickly calculate partial derivatives of multi-variate functions: use the rules of ordinary derivatives with respect to the variable of interest, while considering all other variables as constant. Thus, for example,

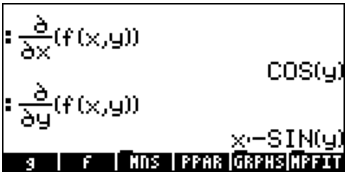
$$\frac{\partial}{\partial x}\left(x\cos(y)\right) = \cos(y), \frac{\partial}{\partial y}\left(x\cos(y)\right) = -x\sin(y),$$

which are the same results as found with the limits calculated earlier. Consider another example,

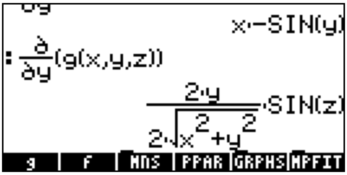
$$\frac{\partial}{\partial x}\left(yx^2 + y^2\right) = 2yx + 0 = 2xy$$

In this calculation we treat y as a constant and take derivatives of the expression with respect to x.

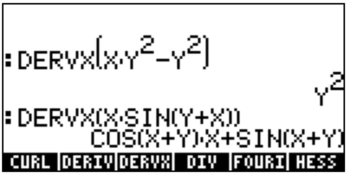
Similarly, you can use the derivative functions in the calculator, e.g., DERVX, DERIV, ∂ (described in detail in Chapter 13) to calculate partial derivatives. Recall that function DERVX uses the CAS default variable VX (typically, 'X'), therefore, with DERVX you can only calculate derivatives with respect to X. Some examples of first-order partial derivatives are shown next:



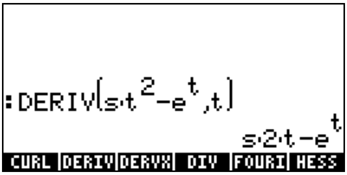
```
: ∂/∂x(f(x,y))
                              COS(y)
: ∂/∂y(f(x,y))
                           x·-SIN(y)
```



```
                           x·-SIN(y)
: ∂/∂y(g(x,y,z))
                  2·y
             ─────────────── ·SIN(z)
             2·√(x^2+y^2)
```



```
:DERVX(X·Y^2-Y^2)
                                 Y^2
:DERVX(X·SIN(Y+X))
      COS(X+Y)·X+SIN(X+Y)
```



```
:DERIV(s·t^2-e^t,t)
                          s·2·t-e^t
```

## Higher-order derivatives

The following second-order derivatives can be defined

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right), \frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right),$$

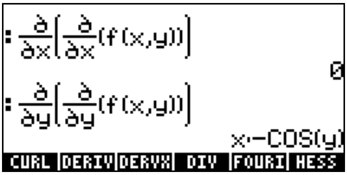
$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right), \frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right)$$

The last two expressions represent cross-derivatives, the partial derivatives signs in the denominator shows the order of derivation. In the left-hand side, the derivation is taking first with respect to x and then with respect to y, and in the right-hand side, the opposite is true. It is important to indicate that, if a function is continuous and differentiable, then

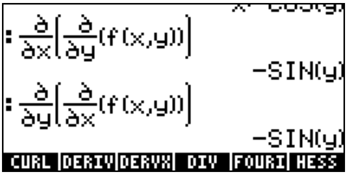
$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial^2 f}{\partial x \partial y}.$$

Third-, fourth-, and higher order derivatives are defined in a similar manner.

To calculate higher order derivatives in the calculator, simply repeat the derivative function as many times as needed. Some examples are shown below:



```
:∂/∂x[∂/∂x(f(x,y))]
                              0
:∂/∂y[∂/∂y(f(x,y))]
                      x·-COS(y)
```



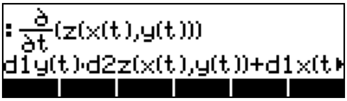
```
                      x·-COS(y)
:∂/∂x[∂/∂y(f(x,y))]
                        -SIN(y)
:∂/∂y[∂/∂x(f(x,y))]
                        -SIN(y)
```

## The chain rule for partial derivatives

Consider the function z = f(x,y), such that x = x(t), y = y(t). The function z actually represents a composite function of *t* if we write it as z = f[x(t),y(t)]. The chain rule for the derivative dz/dt for this case is written as

$$\frac{\partial z}{\partial v} = \frac{\partial z}{\partial x} \cdot \frac{\partial x}{\partial v} + \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial v}$$

To see the expression that the calculator produces for this version of the chain rule use:



```
:∂/∂t(z(x(t),y(t)))
d1y(t)·d2z(x(t),y(t))+d1x(t▸
```

The result is given by d1y(t)·d2z(x(t),y(t))+d1x(t)·d1z(x(y),y(t)). The term d1y(t) is to be interpreted as “the derivative of y(t) with respect to the 1st independent variable, i.e., t”, or d1y(t) = dy/dt. Similarly, d1x(t) = dx/dt. On the other hand, d1z(x(t),y(t)) means “the first derivative of z(x,y) with respect to the first independent variable, i.e., x”, or d1z(x(t),y(t)) = ∂z/∂x. Similarly, d2z(x(t),y(t)) = ∂z/∂y. Thus, the expression above is to be interpreted as:

dz/dt = (dy/dt)·(∂z/∂y) + (dx/dt)·(∂z/∂x).

## Total differential of a function z = z(x,y)

From the last equation, if we multiply by dt, we get the total differential of the function z = z(x,y), i.e., dz = $(\partial z/\partial x)\cdot dx + (\partial z/\partial y)\cdot dy$.

A different version of the chain rule applies to the case in which z = f(x,y), x = x(u,v), y = y(u,v), so that z = f[x(u,v), y(u,v)]. The following formulas represent the chain rule for this situation:

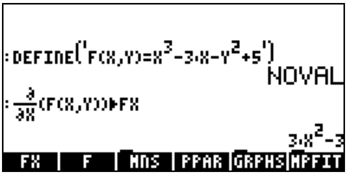
$$\frac{\partial z}{\partial u} = \frac{\partial z}{\partial x}\cdot\frac{\partial x}{\partial u} + \frac{\partial z}{\partial y}\cdot\frac{\partial y}{\partial u}, \qquad \frac{\partial z}{\partial v} = \frac{\partial z}{\partial x}\cdot\frac{\partial x}{\partial v} + \frac{\partial z}{\partial y}\cdot\frac{\partial y}{\partial v}$$

## Determining extrema in functions of two variables

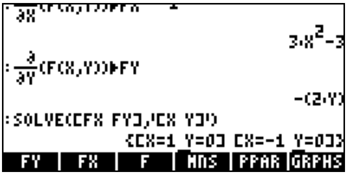
In order for the function z = f(x,y) to have an extreme point (extrema) at $(x_o,y_o)$, its derivatives $\partial f/\partial x$ and $\partial f/\partial y$ must vanish at that point. These are *necessary* conditions. The *sufficient conditions* for the function to have an extreme at point $(x_o,y_o)$ are $\partial f/\partial x = 0$, $\partial f/\partial y = 0$, and $\Delta = (\partial^2 f/\partial x^2)\cdot (\partial^2 f/\partial y^2)-[\partial^2 f/\partial x\partial y]^2 > 0$. The point $(x_o,y_o)$ is a relative maximum if $\partial^2 f/\partial x^2 < 0$, or a relative minimum if $\partial^2 f/\partial x^2 > 0$. The value $\Delta$ is referred to as the discriminant.

If $\Delta = (\partial^2 f/\partial x^2)\cdot (\partial^2 f/\partial y^2)-[\partial^2 f/\partial x\partial y]^2 < 0$, we have a condition known as a *saddle point*, where the function would attain a maximum in x if we were to hold y constant, while, at the same time, attaining a minimum if we were to hold x constant, or vice versa.

Example 1 – Determine the extreme points (if any) of the function $f(X,Y) = X^3-3X-Y^2+5$. First, we define the function f(X,Y), and its derivatives fX(X,Y) = $\partial f/\partial X$, fY(X,Y) = $\partial f/\partial Y$. Then, we solve the equations fX(X,Y) = 0 and fY(X,Y) = 0, simultaneously:

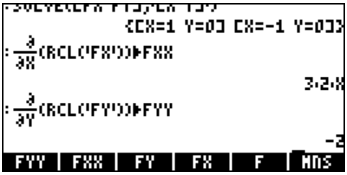


```
:DEFINE('F(X,Y)=X^3-3·X-Y^2+5')
                                NOVAL
:∂/∂X(F(X,Y))▶FX
                                3·X^2-3
```

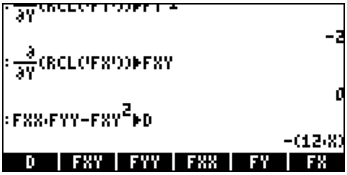


```
                                3·X^2-3
:∂/∂Y(F(X,Y))▶FY
                                -(2·Y)
:SOLVE([FX FY],'[X Y]')
             {[X=1 Y=0] [X=-1 Y=0]}
```

We find critical points at (X,Y) = (1,0), and (X,Y) = (-1,0). To calculate the discriminant, we proceed to calculate the second derivatives, fXX(X,Y) = $\partial^2 f/\partial X^2$, fXY(X,Y) = $\partial^2 f/\partial X/\partial Y$, and fYY(X,Y) = $\partial^2 f/\partial Y^2$.

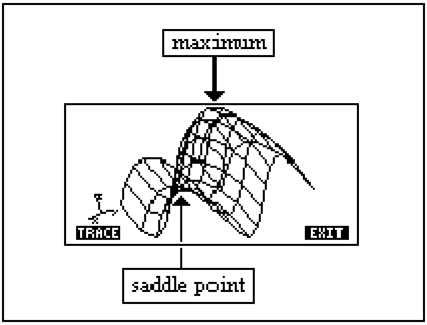


```
                {[X=1 Y=0] [X=-1 Y=0]}
:∂X(RCL('FX'))▶FXX
                                  3·2·X
:∂Y(RCL('FY'))▶FYY
                                     -2
```



```
                                     -2
:∂Y(RCL('FX'))▶FXY
                                      0
:FXX·FYY-FXY²▶D
                                 -(12·X)
```

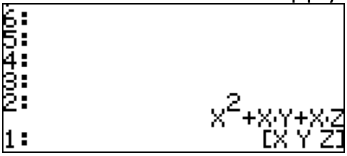
The last result indicates that the discriminant is Δ = -12X, thus, for (X,Y) = (1,0), Δ <0 (saddle point), and for (X,Y) = (-1,0), Δ>0 and $\partial^2 f/\partial X^2 < 0$ (relative maximum). The figure below, produced in the calculator, and edited in the computer, illustrates the existence of these two points:



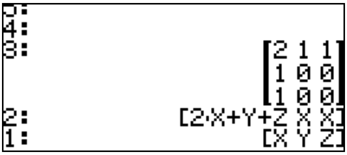
## Using function HESS to analyze extrema

Function HESS can be used to analyze extrema of a function of two variables as shown next. Function HESS, in general, takes as input a function of n independent variables $\phi(x_1, x_2, \ldots, x_n)$, and a vector of the functions ['$x_1$' '$x_2$'…'$x_n$']. Function HESS returns the Hessian matrix of the function $\phi$, defined as the matrix **H** = $[h_{ij}] = [\partial^2\phi/\partial x_i \partial x_j]$, the gradient of the function with respect to the n-variables, **grad** f = [ $\partial\phi/\partial x_1$, $\partial\phi/\partial x_2$ , … $\partial\phi/\partial x_n$], and the list of variables ['$x_1$' '$x_2$'…'$x_n$'].

Applications of function HESS are easier to visualize in the RPN mode. Consider as an example the function $\phi(X,Y,Z) = X^2 + XY + XZ$, we'll apply function HESS to function $\phi$ in the following example. The screen shots show the RPN stack before and after applying function HESS.



```
6:
5:
4:
3:
2:              X²+X·Y+X·Z
1:                 [X Y Z]
```



```
5:
4:
3:                  [2 1 1]
                    [1 0 0]
                    [1 0 0]
2:         [2·X+Y+Z X X]
1:                 [X Y Z]
```

When applied to a function of two variables, the gradient in level 2, when made equal to zero, represents the equations for critical points, i.e., $\partial\phi/\partial x_i = 0$, while the matrix in level 3 represent second derivatives. Thus, the results from the HESS function can be used to analyze extrema in functions of two variables. For example, for the function $f(X,Y) = X^3-3X-Y^2+5$, proceed as follows in RPN mode:

| | |
|---|---|
| 'X^3-3*X-Y^2+5' ENTER ['X','Y'] ENTER | Enter function and variables |
| HESS | Apply function HESS |
| SOLVE | Find critical points |
| EVAL | Decompose vector |
| 's1' STO▸ 's2' STO▸ | Store critical points |

The variables s1 and s2, at this point, contain the vectors ['X=-1','Y=0] and ['X=1','Y=0], respectively. The Hessian matrix is at level 1 at this point.

| | |
|---|---|
| 'H' STO▸ | Store Hessian matrix |
| VAR H S1 SUBST ↱ →NUM | Substitute s1 into H |

The resulting matrix **A** has $a_{11}$ elements $a_{11} = \partial^2\phi/\partial X^2 =$ -6., $a_{22} = \partial^2\phi/\partial X^2 =$ -2., and $a_{12} = a_{21} = \partial^2\phi/\partial X\partial Y =$ 0. The discriminant, for this critical point s1(-1,0) is $\Delta = (\partial^2 f/\partial x^2)\cdot(\partial^2 f/\partial y^2)-[\partial^2 f/\partial x\partial y]^2 = (-6.)(-2.) = 12.0 > 0$. Since $\partial^2\phi/\partial X^2 <0$, point s1 represents a relative maximum.

Next, we substitute the second point, s2, into H:

| | |
|---|---|
| VAR H S2 SUBST ↱ →NUM | Substitute s2 into H |

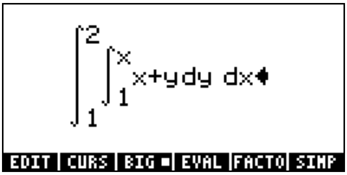
The resulting matrix has elements $a_{11} = \partial^2\phi/\partial X^2 = 6.$, $a_{22} = \partial^2\phi/\partial X^2 = -2.$, and $a_{12} = a_{21} = \partial^2\phi/\partial X\partial Y = 0$. The discriminant, for this critical point s2(1,0) is $\Delta = (\partial^2 f/\partial x^2)\cdot(\partial^2 f/\partial y^2)\text{-}[\partial^2 f/\partial x\partial y]^2 = (6.)(\text{-}2.) = \text{-}12.0 < 0$, indicating a saddle point.

## Multiple integrals

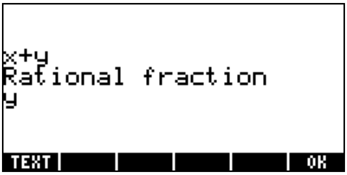
A physical interpretation of an ordinary integral, $\int_a^b f(x)dx$, is the area under the curve y = f(x) and abscissas x = a and x = b. The generalization to three dimensions of an ordinary integral is a double integral of a function f(x,y) over a region R on the x-y plane representing the volume of the solid body contained under the surface f(x,y) above the region R. The region R can be described as R = {a<x<b, f(x)<y<g(x)} or as R = {c<y<d, r(y)<x<s(y)}. Thus, the double integral can be written as

$$\iint_R \phi(x,y)dA = \int_a^b \int_{f(x)}^{g(x)} \phi(x,y)dydx = \int_c^d \int_{r(y)}^{s(y)} \phi(x,y)dydx$$

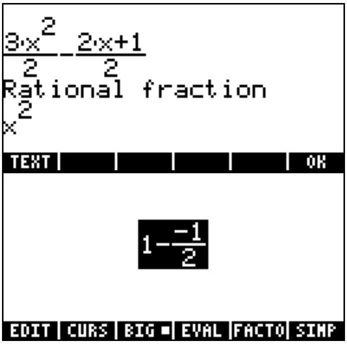
Calculating a double integral in the calculator is straightforward. A double integral can be built in the Equation Writer (see example in Chapter 2). An example follows. This double integral is calculated directly in the Equation Writer by selecting the entire expression and using function EVAL. The result is 3/2. Step-by-step output is possible by setting the Step/Step option in the CAS MODES screen.



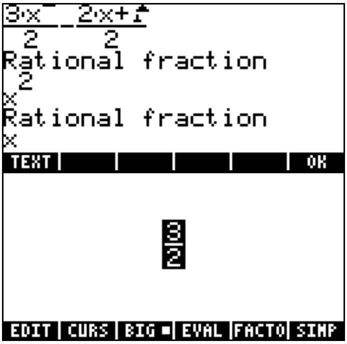
$$\int_1^2 \int_1^x x+y\,dy\,dx$$



```
x+y
Rational fraction
y
```



```
3·x²   2·x+1
──── - ─────
 2       2
Rational fraction
x²
TEXT |  |  |  |  | OK

1 - -1/2

EDIT | CURS | BIG ■ | EVAL | FACTO | SIMP
```



```
3·x²   2·x+1
──── - ─────
 2       2
Rational fraction
x²
Rational fraction
x
TEXT |  |  |  |  | OK

3/2

EDIT | CURS | BIG ■ | EVAL | FACTO | SIMP
```

## Jacobian of coordinate transformation

Consider the coordinate transformation x = x(u,v), y = y(u,v). The Jacobian of this transformation is defined as

$$|J| = \det(J) = \det\begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{pmatrix}.$$

When calculating an integral using such transformation, the expression to use is $\iint_R \phi(x,y)dydx = \iint_{R'} \phi[x(u,v),y(u,v)]\,|J|\,dudv$, where R' is the region R expressed in (u,v) coordinates.

## Double integral in polar coordinates

To transform from polar to Cartesian coordinates we use x(r,θ) = r cos θ, and y(r, θ) = r sin θ. Thus, the Jacobian of the transformation is

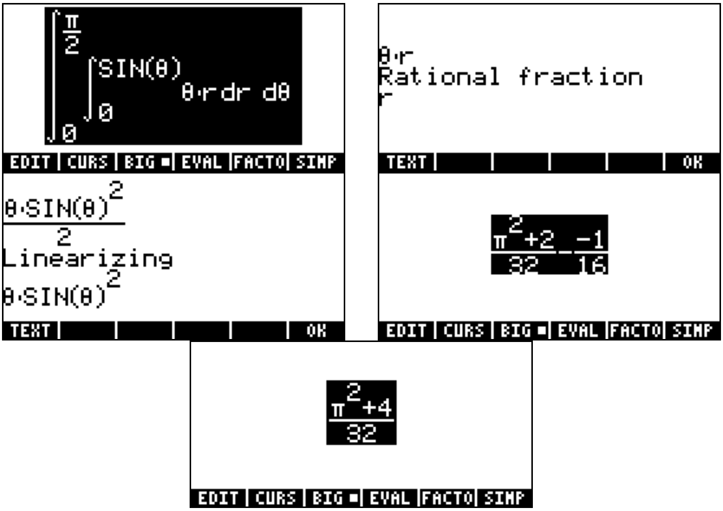
$$|J| = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -r\cdot\sin(\theta) \\ \sin(\theta) & r\cdot\cos(\theta) \end{vmatrix} = r$$

With this result, integrals in polar coordinates are written as

$$\iint_{R'} \phi(r,\theta)dA = \int_{\alpha}^{\beta}\int_{f(\theta)}^{g(\theta)} \phi(r,\theta) r dr d\theta$$

where the region R′ in polar coordinates is R′ = {$\alpha < \theta < \beta$, $f(\theta) < r < g(\theta)$}.

Double integrals in polar coordinates can be entered in the calculator, making sure that the Jacobian |J| = r is included in the integrand. The following is an example of a double integral calculated in polar coordinates, shown step-by-step:



$$\int_0^{\frac{\pi}{2}}\int_0^{SIN(\theta)} \theta\cdot r dr\ d\theta$$

θ·r
Rational fraction
r

$$\frac{\theta\cdot SIN(\theta)^2}{2}$$

Linearizing

$\theta\cdot SIN(\theta)^2$

$$\frac{\pi^2+2}{32}-\frac{1}{16}$$

$$\frac{\pi^2+4}{32}$$

# Chapter 15
# Vector Analysis Applications

In this Chapter we present a number of functions from the CALC menu that apply to the analysis of scalar and vector fields. The CALC menu was presented in detail in Chapter 13. In particular, in the DERIV&INTEG menu we identified a number of functions that have applications in vector analysis, namely, CURL, DIV, HESS, LAPL. For the exercises in this Chapter, change your angle measure to radians.

## Definitions

A function defined in a region of space such as φ(x,y,z) is known as a scalar field, examples are temperature, density, and voltage near a charge. If the function is defined by a vector, i.e., **F**(x,y,z) = f(x,y,z)**i**+g(x,y,z)**j**+h(x,y,z)**k**, it is referred to as a vector field.

The following operator, referred to as the 'del' or 'nabla' operator, is a vector-based operator that can be applied to a scalar or vector function:

$$\nabla[\ ] = i\cdot\frac{\partial}{\partial x}[\ ] + j\cdot\frac{\partial}{\partial y}[\ ] + k\cdot\frac{\partial}{\partial z}[\ ]$$

When this operator is applied to a scalar function we can obtain the gradient of the function, and when applied to a vector function we can obtain the divergence and the curl of that function. A combination of gradient and divergence produces another operator, called the Laplacian of a scalar function. These operations are presented next.

## Gradient and directional derivative

The gradient of a scalar function φ(x,y,z) is a vector function defined by

$$grad\phi = \nabla\phi = i\cdot\frac{\partial\phi}{\partial x} + j\cdot\frac{\partial\phi}{\partial y} + k\cdot\frac{\partial\phi}{\partial z}$$

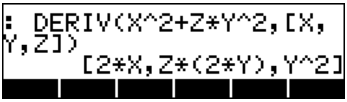
The dot product of the gradient of a function with a given unit vector represents the rate of change of the function along that particular vector. This rate of change is called the directional derivative of the function, $D_u\phi(x,y,z)$ = **u**•∇φ.

At any particular point, the maximum rate of change of the function occurs in the direction of the gradient, i.e., along a unit vector **u** = ∇φ/|∇φ|.

The value of that directional derivative is equal to the magnitude of the gradient at any point $D_{max}\phi(x,y,z) = \nabla\phi \bullet \nabla\phi/|\nabla\phi| = |\nabla\phi|$

The equation φ(x,y,z) = 0 represents a surface in space. It turns out that the gradient of the function at any point on this surface is normal to the surface. Thus, the equation of a plane tangent to the curve at that point can be found by using a technique presented in Chapter 9.

The simplest way to obtain the gradient is by using function DERIV, available in the CALC menu, e.g.,



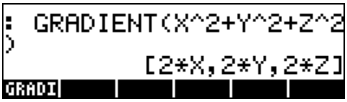
```
: DERIV(X^2+Z*Y^2,[X,
Y,Z])
      [2*X,Z*(2*Y),Y^2]
```

## A program to calculate the gradient

The following program, which you can store into variable GRADIENT, uses function DERIV to calculate the gradient of a scalar function of X,Y,Z. Calculations for other base variables will not work. If you work frequently in the (X,Y,Z) system, however, this function will facilitate calculations:

```
<<  X  Y  Z  3 →ARRY DERIV >>
```

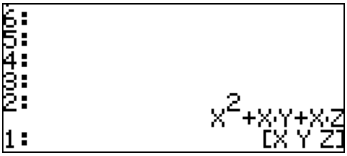
Type the program while in RPN mode. After switching to ALG mode, you can call the function GRADIENT as in the following example:



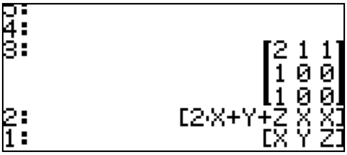
```
: GRADIENT(X^2+Y^2+Z^2
)
            [2*X,2*Y,2*Z]
```

## Using function HESS to obtain the gradient

The function HESS can be used to obtain the gradient of a function as shown next. As indicated in Chapter 14, function HESS takes as input a function of n independent variables $\phi(x_1, x_2, \ldots, x_n)$, and a vector of the functions [‘$x_1$’ ‘$x_2$’…’$x_n$’]. Function HESS returns the Hessian matrix of the function $\phi$, defined as the matrix $\mathbf{H} = [h_{ij}] = [\partial\phi/\partial x_i\partial x_j]$, the gradient of the function with respect to the n-variables, **grad** f = [ $\partial\phi/\partial x_1$, $\partial\phi/\partial x_2$ , … $\partial\phi/\partial x_n$], and the list of variables [‘$x_1$’ ‘$x_2$’…’$x_n$’]. Consider as an example the function $\phi(X,Y,Z) = X^2 + XY + XZ$, we’ll apply function HESS to this scalar field in the following example in RPN mode:



```
6:
5:
4:
3:
2:                 X^2+X·Y+X·Z
1:                     [X Y Z]
```



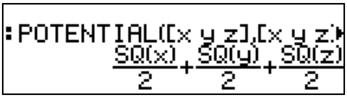
```
5:
4:
3:                      [2 1 1]
                        [1 0 0]
                        [1 0 0]
2:           [2·X+Y+Z X X]
1:                 [X Y Z]
```

Thus, the gradient is [2X+Y+Z, X, X]. Alternatively, one can use function DERIV as follows: DERIV(X^2+X*Y+X*Z,[X,Y,Z]), to obtain the same result.

## Potential of a gradient

Given the vector field, $\mathbf{F}(x,y,z) = f(x,y,z)\mathbf{i}+g(x,y,z)\mathbf{j}+h(x,y,z)\mathbf{k}$, if there exists a function $\phi(x,y,z)$, such that $f = \partial\phi/\partial x$, $g = \partial\phi/\partial y$, and $h = \partial\phi/\partial z$, then $\phi(x,y,z)$ is referred to as the potential function for the vector field **F**. It follows that $\mathbf{F} = \text{grad } \phi = \nabla\phi$.

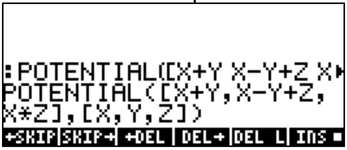
The calculator provides function POTENTIAL, available through the command catalog (↱ CAT ), to calculate the potential function of a vector field, if it exists. For example, if $\mathbf{F}(x,y,z) = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, applying function POTENTIAL we find:



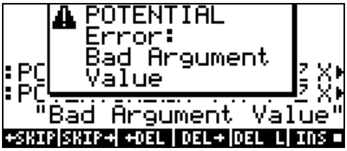
```
:POTENTIAL([x y z],[x y z]▸
          SQ(x)   SQ(y)   SQ(z)
          ----- + ----- + -----
            2       2       2
```

Since function SQ(x) represents $x^2$, this results indicates that the potential function for the vector field $\mathbf{F}(x,y,z) = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, is $\phi(x,y,z) = (x^2+y^2+z^2)/2$.

Notice that the conditions for the existence of $\phi(x,y,z)$, namely, $f = \partial\phi/\partial x$, $g = \partial\phi/\partial y$, and $h = \partial\phi/\partial z$, are equivalent to the conditions: $\partial f/\partial y = \partial g/\partial x$, $\partial f/\partial z = \partial h/\partial x$, and $\partial g/\partial z = \partial h/\partial y$. These conditions provide a quick way to determine if the vector field has an associated potential function. If one of the conditions $\partial f/\partial y = \partial g/\partial x$, $\partial f/\partial z = \partial h/\partial x$, $\partial g/\partial z = \partial h/\partial y$, fails, a potential function ϕ(x,y,z) does not exist. In such case, function POTENTIAL returns an error message. For example, the vector field **F**(x,y,z) = (x+y)**i** + (x-y+z)**j** + xz**k**, does not have a potential function associated with it, since, ∂f/∂z ≠ ∂h/∂x. The calculator response in this case is shown below:



```
:POTENTIAL([X+Y X-Y+Z X▸
POTENTIAL([X+Y,X-Y+Z,
X*Z],[X,Y,Z])
```



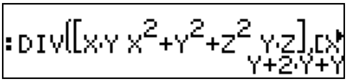
```
POTENTIAL
Error:
Bad Argument
Value
:PC                 Z X▸
:PC                 Z X▸
"Bad Argument Value"
```

## Divergence

The divergence of a vector function, **F**(x,y,z) = f(x,y,z)**i**+g(x,y,z)**j**+h(x,y,z)**k**, is defined by taking a "dot-product" of the del operator with the function, i.e.,

$$divF = \nabla \bullet F = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z}$$

Function DIV can be used to calculate the divergence of a vector field. For example, for **F**(X,Y,Z) = [XY,X$^2$+Y$^2$+Z$^2$,YZ], the divergence is calculated, in ALG mode, as follows:



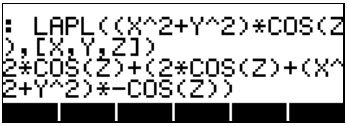
```
:DIV([X·Y X²+Y²+Z² Y·Z],[X▸
                    Y+2·Y+Y
```

## Laplacian

The divergence of the gradient of a scalar function produces an operator called the Laplacian operator. Thus, the Laplacian of a scalar function ϕ(x,y,z) is given by

$$\nabla^2 \phi = \nabla \bullet \nabla \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial x^2}$$

The partial differential equation ∇$^2$ϕ = 0 is known as Laplace's equation. Function LAPL can be used to calculate the Laplacian of a scalar function. For example, to calculate the Laplacian of the function ϕ(X,Y,Z) = (X$^2$+Y$^2$)cos(Z), use:



```
: LAPL((X^2+Y^2)*COS(Z
),[X,Y,Z])
2*COS(Z)+(2*COS(Z)+(X^
2+Y^2)*-COS(Z))
```

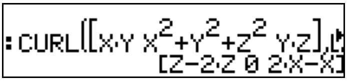
# Curl

The curl of a vector field **F**(x,y,z) = f(x,y,z)**i**+g(x,y,z)**j**+h(x,y,z)**k**, is defined by a "cross-product" of the del operator with the vector field, i.e.,

$$curl\mathbf{F} = \nabla \times \mathbf{F} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x}[\ ] & \frac{\partial}{\partial y}[\ ] & \frac{\partial}{\partial z}[\ ] \\ f(x,y,z) & g(x,y,z) & h(x,y,z) \end{vmatrix}$$

$$= \mathbf{i}\left(\frac{\partial h}{\partial y} - \frac{\partial g}{\partial z}\right) + \mathbf{j}\left(\frac{\partial f}{\partial z} - \frac{\partial h}{\partial x}\right) + \mathbf{k}\left(\frac{\partial h}{\partial y} - \frac{\partial g}{\partial z}\right)$$

The curl of vector field can be calculated with function CURL. For example, for the function **F**(X,Y,Z) = [XY,X$^2$+Y$^2$+Z$^2$,YZ], the curl is calculated as follows:



```
:CURL([[X·Y X²+Y²+Z² Y·Z],[
                 [Z-2·Z 0 2·X-X]
```

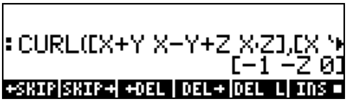
## Irrotational fields and potential function

In an earlier section in this chapter we introduced function POTENTIAL to calculate the potential function φ(x,y,z) for a vector field, **F**(x,y,z) = f(x,y,z)**i**+ g(x,y,z)**j**+ h(x,y,z)**k**, such that **F** = grad φ = ∇φ. We also indicated that the conditions for the existence of φ, were: ∂f/∂y = ∂g/∂x, ∂f/∂z = ∂h/∂x, and ∂g/∂z = ∂h/∂y. These conditions are equivalent to the vector expression

$$\text{curl } \mathbf{F} = \nabla\times\mathbf{F} = 0.$$

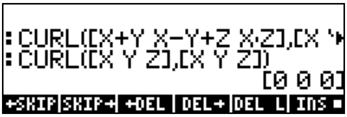
A vector field **F**(x,y,z), with zero curl, is known as an irrotational field. Thus, we conclude that a potential function φ(x,y,z) always exists for an irrotational field **F**(x,y,z).

As an example, in an earlier example we attempted to find a potential function for the vector field **F**(x,y,z) = (x+y)**i** + (x-y+z)**j** + xz**k**, and got an error message back from function POTENTIAL. To verify that this is a rotational field (i.e., ∇×**F** ≠ 0), we use function CURL on this field:



```
:CURL([X+Y X-Y+Z X·Z],[X '▸
                    [-1 -Z 0]
```

On the other hand, the vector field **F**(x,y,z) = **xi** + y**j** + z**k**, is indeed irrotational as shown below:

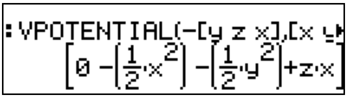


```
:CURL([X+Y X-Y+Z X·Z],[X '▸
:CURL([X Y Z],[X Y Z])
                      [0 0 0]
```

# Vector potential

Given a vector field **F**(x,y,z) = f(x,y,z)**i**+g(x,y,z)**j**+h(x,y,z)**k**, if there exists a vector function Φ(x,y,z) = ϕ(x,y,z)**i**+ψ(x,y,z)**j**+η(x,y,z)**k**, such that **F** = curl Φ = ∇× Φ, then function Φ(x,y,z) is referred to as the vector potential of **F**(x,y,z).

The calculator provides function VPOTENTIAL, available through the command catalog (↱ CAT ), to calculate the vector potential, Φ(x,y,z), given the vector field, **F**(x,y,z) = f(x,y,z)**i**+g(x,y,z)**j**+h(x,y,z)**k**. For example, given the vector field, **F**(x,y,z) = -(y**i**+z**j**+x**k**), function VPOTENTIAL produces

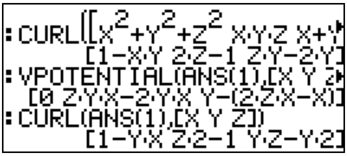


```
:VPOTENTIAL(-[y z x],[x y▸
```

$$\left[0\ -\left(\frac{1}{2}\cdot x^2\right)\ -\left(\frac{1}{2}\cdot y^2\right)+z\cdot x\right]$$

i.e., Φ(x,y,z) = $-x^2/2$**j** + ($-y^2/2+zx$)**k**.

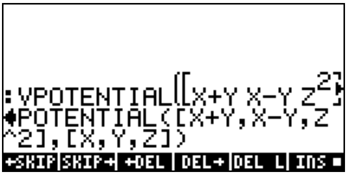
It should be indicated that there is more than one possible vector potential functions Φ for a given vector field **F**. For example, the following screen shot shows that the curl of the vector function $\Phi_1$ = [$X^2+Y^2+Z^2$,XYZ,X+Y+Z] is the vector **F** = ∇× $\Phi_2$ = [1-XY,2Z-1,ZY-2Y]. Application of function VPOTENTIAL produces the vector potential function $\Phi_2$ = [0, ZYX-2YX, Y-(2ZX-X)], which is different from $\Phi_1$. The last command in the screen shot shows that indeed **F** = $\nabla\times\ \Phi_2$. Thus, a vector potential function is not uniquely determined.



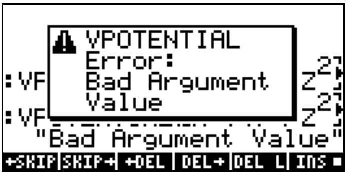
```
:CURL([[X^2+Y^2+Z^2 X·Y·Z X+Y▸
      [1-X·Y 2·Z-1 Z·Y-2·Y]
:VPOTENTIAL(ANS(1),[X Y Z▸
  [0 Z·Y·X-2·Y·X Y-(2·Z·X-X)]
:CURL(ANS(1),[X Y Z])
      [1-Y·X Z·2-1 Y·Z-Y·2]
```

The components of the given vector field, **F**(x,y,z) = f(x,y,z)**i**+g(x,y,z)**j** +h(x,y,z)**k**, and those of the vector potential function, $\Phi$(x,y,z) = $\phi$(x,y,z)**i**+$\psi$(x,y,z)**j**+$\eta$(x,y,z)**k**, are related by f = $\partial\eta/\partial y$ - $\partial\psi/\partial x$, g = $\partial\phi/\partial z$ - $\partial\eta/\partial x$, and h = $\partial\psi/\partial x$ - $\partial\phi/\partial y$.

A condition for function $\Phi$(x,y,z) to exists is that div **F** = $\nabla$•**F** = 0, i.e., $\partial f/\partial x$ + $\partial g/\partial y$ + $\partial f/\partial z$ = 0. Thus, if this condition is not satisfied, the vector potential function $\Phi$(x,y,z) does not exist. For example, given **F** = [X+Y,X-Y,Z^2], function VPOTENTIAL returns an error message, since function F does not satisfy the condition $\nabla$•**F** = 0:

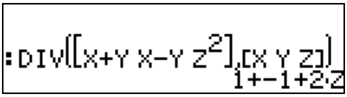


```
:VPOTENTIAL([[X+Y X-Y Z^2]▸
◂POTENTIAL([X+Y,X-Y,Z
^2],[X,Y,Z])
```



```
VPOTENTIAL
Error:
Bad Argument
Value
"Bad Argument Value"
```

The condition $\nabla$•**F** ≠ 0 is verified in the following screen shot:



```
:DIV([[X+Y X-Y Z^2],[X Y Z])
                    1+-1+2·Z
```

# Chapter 16
# Differential Equations

In this Chapter we present examples of solving ordinary differential equations (ODE) using calculator functions. A differential equation is an equation involving derivatives of the independent variable. In most cases, we seek the dependent function that satisfies the differential equation.

## Basic operations with differential equations

In this section we present some uses of the calculator for entering, checking and visualizing the solution of ODEs.

### Entering differential equations

The key to using differential equations in the calculator is typing in the derivatives in the equation. The easiest way to enter a differential equation is to type it in the equation writer. For example, to type the following ODE: (x-1)·(dy/dx)$^2$ + 2·x·y = e$^{-x}$ sin x, use:

↱ EQW ↰ () ALPHA ↰ X − 1 ▶ ▶ ▶ × ↱ ∂ ALPHA ↰ X
▶ ALPHA ↰ Y ↰ () ALPHA ↰ X ▶ ▶ Y$^x$ 2 ▶ ▶ + 2 ×
ALPHA ↰ X × ALPHA ↰ Y ↰ () ALPHA ↰ X ▶ ▶ ▶ ▶
↱ = ↰ e$^x$ ALPHA ↰ X ▶ × SIN ALPHA ↰ X ENTER

The derivative dy/dx is represented by `∂x(y(x)) or by d1y(x).` For solution or calculation purposes, you need to specify y(x) in the expression, i.e., the dependent variable must include its independent variable(s) in any derivative in the equation.

You can also type an equation directly into the stack by using the symbol ∂ in the derivatives. For example, to type the following ODE involving second-order derivatives: d$^2$u/dx$^2$ + 3·u·(du/dx) + u$^2$ = 1/x, directly into the stack, use:

' ↱ ∂ ALPHA ↰ X ↰ () ↱ ∂ ALPHA ↰ X ↰ () ALPHA ↰ U
↰ () ALPHA ↰ X ▶ ▶ ▶ + 3 × ALPHA ↰ U ↰ ()
ALPHA ↰ X ▶ × ↱ ∂ ALPHA ↰ X ↰ () ALPHA ↰ U ↰ ()
ALPHA ↰ X ▶ ▶ + ALPHA ↰ U ↰ () ALPHA ↰ X ▶ Y$^x$ 2
↱ = 1 ÷ ALPHA ↰ X ENTER

The result is `'∂x(∂x(u(x)))+3*u(x)*∂x(u(x))+u^2=1/x '`. This format shows up in the screen when the _Textbook option in the display setting (MODE DISP) is not selected. Press ▼ to see the equation in the Equation Writer.

An alternative notation for derivatives typed directly in the stack is to use 'd1' for the derivative with respect to the first independent variable, 'd2' for the derivative with respect to the second independent variable, etc. A second-order derivative, e.g., $d^2x/dt^2$, where x = x(t), would be written as 'd1d1x(t)', while $(dx/dt)^2$ would be written 'd1x(t)^2'. Thus, the PDE $\partial^2y/\partial t^2 - g(x,y)\cdot(\partial^2y/\partial x^2)^2 = r(x,y)$, would be written, using this notation, as 'd2d2y(x,t)-g(x,y)*d1d1y(x,t)^2=r(x,y)'.

The notation using 'd' and the order of the independent variable is the notation preferred by the calculator when derivatives are involved in a calculation. For example, using function DERIV, in ALG mode, as shown next DERIV('x*f(x,t)+g(t,y) = h(x,y,t)',t), produces the following expression: `'x*d2f(x,t)+d1g(t,y)=d3h(x,y,t)'`. Translated to paper, this expression represents the partial differential equation $x\cdot(\partial f/\partial t) + \partial g/\partial t = \partial h/\partial t$.

Because the order of the variable t is different in f(x,t), g(t,y), and h(x,y,t), derivatives with respect to t have different indices, i.e., d2f(x,t), d1g(t,y), and d3h(x,y,t). All of them, however, represent derivatives with respect to the same variable.

Expressions for derivatives using the order-of-variable index notation do not translate into derivative notation in the equation writer, as you can check by pressing ▼ while the last result is in stack level 1. However, the calculator understands both notations and operates accordingly regarding of the notation used.

## Checking solutions in the calculator

To check if a function satisfy a certain equation using the calculator, use function SUBST (see Chapter 5) to replace the solution in the form 'y = f(x)' or 'y = f(x,t)', etc., into the differential equation. You may need to simplify the result by using function EVAL to verify the solution. For example, to check that u = A sin $\omega_o t$ is a solution of the equation $d^2u/dt^2 + \omega_o^2\cdot u = 0$, use the following:

In ALG mode:

SUBST(‘∂t(∂t(u(t)))+ ω0^2*u(t) = 0’,‘u(t)=A*SIN (ω0*t)’) ENTER

EVAL(ANS(1)) ENTER

In RPN mode:

‘∂t(∂t(u(t)))+ ω0^2*u(t) = 0’ ENTER ‘u(t)=A*SIN (ω0*t)’ ENTER

SUBST EVAL

The result is ‘0=0’.

For this example, you could also use: ‘∂t(∂t(u(t))))+ ω0^2*u(t) = 0’ to enter the differential equation.

## Slope field visualization of solutions

Slope field plots, introduced in Chapter 12, are used to visualize the solutions to a differential equation of the form dy/dx = f(x,y). A slope field plot shows a number of segments tangential to the solution curves, y = f(x). The slope of the segments at any point (x,y) is given by dy/dx = f(x,y), evaluated at any point (x,y), represents the slope of the tangent line at point (x,y).

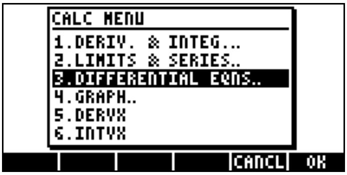
Example 1 -- Trace the solution to the differential equation y’ = f(x,y) = sin x cos y, using a slope field plot. To solve this problem, follow the instructions in Chapter 12 for *slopefield* plots.

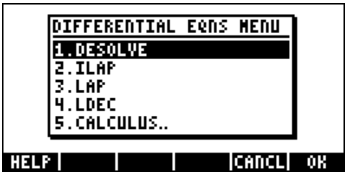
If you could reproduce the slope field plot in paper, you can trace by hand lines that are tangent to the line segments shown in the plot. This lines constitute lines of y(x,y) = constant, for the solution of y’ = f(x,y). Thus, slope fields are useful tools for visualizing particularly difficult equations to solve.

In summary, slope fields are graphical aids to sketch the curves y = g(x) that correspond to solutions of the differential equation dy/dx = f(x,y).

# The CALC/DIFF menu

The DIFFERENTIAL EQNS.. sub-menu within the CALC (← CALC ) menu provides functions for the solution of differential equations. The menu is listed below with system flag 117 set to CHOOSE boxes:





These functions are briefly described next. They will be described in more detail in later parts of this Chapter.

DESOLVE: Differential Equation SOLVEr, provides a solution if possible
ILAP: Inverse LAPlace transform, $L^{-1}[F(s)] = f(t)$
LAP: LAPlace transform, $L[f(t)]=F(s)$
LDEC: solves Linear Differential Equations with Constant coefficients, including systems of differential equations with constant coefficients

# Solution to linear and non-linear equations

An equation in which the dependent variable and all its pertinent derivatives are of the first degree is referred to as a linear differential equation. Otherwise, the equation is said to be non-linear. Examples of linear differential equations are: $d^2x/dt^2 + \beta\cdot(dx/dt) + \omega_o\cdot x = A \sin \omega_f t$, and $\partial C/\partial t + u\cdot(\partial C/\partial x) = D\cdot(\partial^2 C/\partial x^2)$.

An equation whose right-hand side (not involving the function or its derivatives) is equal to zero is called a homogeneous equation. Otherwise, it is called non-homogeneous. The solution to the homogeneous equation is known as a general solution. A particular solution is one that satisfies the non-homogeneous equation.

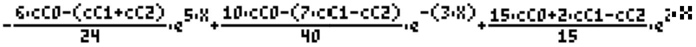
## Function LDEC

The calculator provides function LDEC (Linear Differential Equation Command) to find the general solution to a linear ODE of any order with constant coefficients, whether it is homogeneous or not. This function requires you to provide two pieces of input:

- the right-hand side of the ODE
- the characteristic equation of the ODE

Both of these inputs must be given in terms of the default independent variable for the calculator's CAS (typically 'X'). The output from the function is the general solution of the ODE. The function LDEC is available through in the CALC/DIFF menu. The examples are shown in the RPN mode, however, translating them to the ALG mode is straightforward.

Example 1 – To solve the homogeneous ODE: $d^3y/dx^3$-4·($d^2y/dx^2$)-11·(dy/dx)+30·y = 0, enter: `0` ENTER `'X^3-4*X^2-11*X+30'` ENTER `LDEC`. The solution is:



$$-\frac{6\cdot cC0-(cC1+cC2)}{24}\cdot e^{5\cdot X}+\frac{10\cdot cC0-(7\cdot cC1-cC2)}{40}\cdot e^{-(3\cdot X)}+\frac{15\cdot cC0+2\cdot cC1-cC2}{15}\cdot e^{2\cdot X}$$

where cC0, cC1, and cC2 are constants of integration. While this result seems very complicated, it can be simplified if we take

K1 = (10*cC0-(7+cC1-cC2))/40, K2 = -(6*cC0-(cC1+cC2))/24,

and

K3 = (15*cC0+(2*cC1-cC2))/15.

Then, the solution is

$$y = K_1 \cdot e^{-3x} + K_2 \cdot e^{5x} + K_3 \cdot e^{2x}.$$

The reason why the result provided by LDEC shows such complicated combination of constants is because, internally, to produce the solution, LDEC utilizes Laplace transforms (to be presented later in this chapter), which transform the solution of an ODE into an algebraic solution. The combination of constants result from factoring out the exponential terms after the Laplace transform solution is obtained.

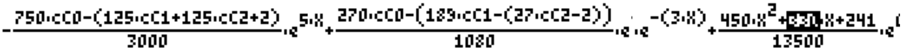
Example 2 – Using the function LDEC, solve the non-homogeneous ODE:

$$d^3y/dx^3\text{-}4\cdot(d^2y/dx^2)\text{-}11\cdot(dy/dx)+30\cdot y = x^2.$$

Enter:

```
'X^2' ENTER 'X^3-4*X^2-11*X+30' ENTER LDEC
```

The solution, shown partially here in the Equation Writer, is:



$$-\frac{750\cdot cC0-(125\cdot cC1+125\cdot cC2+2)}{3000}\cdot e^{5\cdot X}+\frac{270\cdot cC0-(189\cdot cC1-(27\cdot cC2-2))}{1080}\cdot e\cdot e^{-(3\cdot X)}+\frac{450\cdot X^2+330\cdot X+241}{13500}\cdot e^{(}$$

Replacing the combination of constants accompanying the exponential terms with simpler values, the expression can be simplified to $y = K_1\cdot e^{-3x} + K_2\cdot e^{5x} + K_3\cdot e^{2x} + (450\cdot x^2+330\cdot x+241)/13500$.

We recognize the first three terms as the general solution of the homogeneous equation (see Example 1, above). If $y_h$ represents the solution to the homogeneous equation, i.e., $y_h = K_1\cdot e^{-3x} + K_2\cdot e^{5x} + K_3\cdot e^{2x}$. You can prove that the remaining terms in the solution shown above, i.e., $y_p = (450\cdot x^2+330\cdot x+241)/13500$, constitute a particular solution of the ODE.

**Note**: This result is general for all non-homogeneous linear ODEs, i.e., given the solution of the homogeneous equation, $y_h(x)$, the solution of the corresponding non-homogeneous equation, $y(x)$, can be written as

$$y(x) = y_h(x) + y_p(x),$$

where $y_p(x)$ is a particular solution to the ODE.

To verify that $y_p = (450\cdot x^2+330\cdot x+241)/13500$, is indeed a particular solution of the ODE, use the following:

```
'd1d1d1Y(X)-4*d1d1Y(X)-11*d1Y(X)+30*Y(X) = X^2' ENTER
        'Y(X)=(450*X^2+330*X+241)/13500' ENTER
                    SUBST  EVAL
```

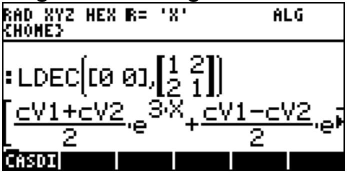
Allow the calculator about ten seconds to produce the result: ‘X^2 = X^2’.

Example 3 - Solving a system of linear differential equations with constant coefficients.

Consider the system of linear differential equations:

$$x_1'(t) + 2x_2'(t) = 0,$$
$$2x_1'(t) + x_2'(t) = 0.$$

In algebraic form, this is written as: $\mathbf{A}\cdot\mathbf{x}'(t) = 0$, where $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$. The system can be solved by using function LDEC with arguments [0,0] and matrix A, as shown in the following screen using ALG mode:

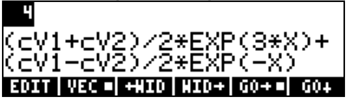


RAD XYZ HEX R= 'X' ALG
{HOME}

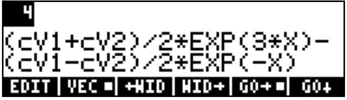
$:\mathrm{LDEC}\left([0\ 0],\begin{bmatrix}1 & 2\\ 2 & 1\end{bmatrix}\right)$

$\left[\frac{cV1+cV2}{2}\cdot e^{3\cdot X}+\frac{cV1-cV2}{2}\cdot e\right.$

The solution is given as a vector containing the functions [$x_1(t)$, $x_2(t)$]. Pressing ▼ will trigger the Matrix Writer allowing the user to see the two components of the vector. To see all the details of each component, press the EDIT soft menu key. Verify that the components are:



```
4
(cV1+cV2)/2*EXP(3*X)+
(cV1-cV2)/2*EXP(-X)
```



```
4
(cV1+cV2)/2*EXP(3*X)-
(cV1-cV2)/2*EXP(-X)
```

## Function DESOLVE

The calculator provides function DESOLVE (Differential Equation SOLVEr) to solve certain types of differential equations. The function requires as input the differential equation and the unknown function, and returns the solution to the equation if available. You can also provide a vector containing the differential equation and the initial conditions, instead of only a differential equation, as input to DESOLVE. The function DESOLVE is available in the CALC/DIFF menu. Examples of DESOLVE applications are shown below using RPN mode.

Example 1 – Solve the first-order ODE:

$$dy/dx + x^2 \cdot y(x) = 5.$$

In the calculator use:

`'d1y(x)+x^2*y(x)=5'` ENTER `'y(x)'` ENTER `DESOLVE`

The solution provided is {'y = (INT(5*EXP(xt^3/3),xt,x)+cC0)*1/EXP(x^3/3))' }, i.e.,

$$y(x) = \exp(-x^3/3)\cdot\left(\int 5\cdot\exp(x^3/3)\cdot dx + cC_0\right)$$

## The variable ODETYPE

You will notice in the soft-menu key labels a new variable called ODETY (ODETYPE). This variable is produced with the call to the DESOL function and holds a string showing the type of ODE used as input for DESOLVE. Press ODETY to obtain the string "`1st order linear`".

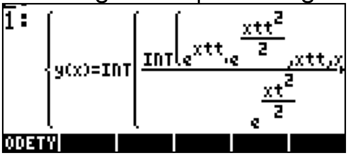
Example 2 -- Solve the second-order ODE:

$$d^2y/dx^2 + x\ (dy/dx) = \exp(x).$$

In the calculator use:

`'d1d1y(x)+x*d1y(x) = EXP(x)'` ENTER `'y(x)'` ENTER `DESOLVE`

The result is an expression having two implicit integrations, namely,



1: $\left\{ y(x)=\mathrm{INT}\left( \frac{\mathrm{INT}\left(e^{xtt}\cdot e^{\frac{xtt^2}{2}},xtt,x\right.}{e^{\frac{xt^2}{2}}} \right. \right.$

For this particular equation, however, we realize that the left-hand side of the equation represents d/dx(x dy/dx), thus, the ODE is now written:

$$d/dx(x\ dy/dx\ ) = \exp x,$$

and

$$x\ dy/dx = \exp x + C.$$

Next, we can write

$$dy/dx = (C + \exp x)/x = C/x + e^x/x.$$

In the calculator, you may try to integrate:

‘d1y(x) = (C + EXP(x))/x’ ENTER ‘y(x)’ ENTER DESOLVE

The result is { ‘y(x) = INT((EXP(xt)+C)/xt,xt,x)+C0’ }, i.e.,

$$y(x) = \int \cdot \frac{e^x + C}{x} dx + C_0$$

Performing the integration by hand, we can only get it as far as:

$$y(x) = \int \cdot \frac{e^x}{x} dx + C \cdot \ln x + C_0$$

because the integral of exp(x)/x is not available in closed form.

<u>Example 3</u> – Solving an equation with initial conditions. Solve

$$d^2y/dt^2 + 5y = 2\cos(t/2),$$

with initial conditions

$$y(0) = 1.2,\ y'(0) = -0.5.$$

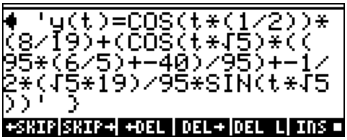
In the calculator, use:

[‘d1d1y(t)+5*y(t) = 2*COS(t/2)’ ‘y(0) = 6/5’ ‘d1y(0) = -1/2’] ENTER
‘y(t)’ ENTER
DESOLVE

Notice that the initial conditions were changed to their Exact expressions, ‘y(0) = 6/5’, rather than ‘y(0)=1.2’, and ‘d1y(0) = -1/2’, rather than, ‘d1y(0) = -0.5’. Changing to these Exact expressions facilitates the solution.

**Note**: To obtain fractional expressions for decimal values use function →Q (See Chapter 5).

The solution is:



```
♦ 'y(t)=COS(t*(1/2))*
(8/19)+(COS(t*√5)*((
95*(6/5)+-40)/95)+-1/
2*(√5*19)/95*SIN(t*√5
))' }
```

Press EVAL EVAL to simplify the result to
‘y(t) = -((19*√5*SIN(√5*t)-(148*COS(√5*t)+80*COS(t/2)))/190)’.

Press VAR ODETY to get the string “`Linear w/ cst coeff`” for the ODE type in this case.

# Laplace Transforms

The Laplace transform of a function f(t) produces a function F(s) in the image domain that can be utilized to find the solution of a linear differential equation involving f(t) through algebraic methods. The steps involved in this application are three:

1. Use of the Laplace transform converts the linear ODE involving f(t) into an algebraic equation.
2. The unknown F(s) is solved for in the image domain through algebraic manipulation.
3. An inverse Laplace transform is used to convert the image function found in step 2 into the solution to the differential equation f(t).

## Definitions

The Laplace transform for function f(t) is the function F(s) defined as

$$\mathrm{L}\{f(t)\} = F(s) = \int_0^{\infty} f(t)\cdot e^{-st}dt.$$

The image variable s can be, and it generally is, a complex number.

Many practical applications of Laplace transforms involve an original function f(t) where t represents time, e.g., control systems in electric or hydraulic

circuits. In most cases one is interested in the system response after time t>0, thus, the definition of the Laplace transform, given above, involves an integration for values of t larger than zero.
The inverse Laplace transform maps the function F(s) onto the original function f(t) in the time domain, i.e., L $^{-1}${F(s)} = f(t).

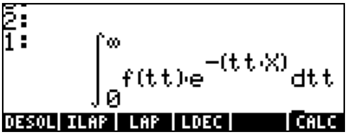
The convolution integral or convolution product of two functions f(t) and g(t), where g is shifted in time, is defined as

$$(f * g)(t) = \int_0^t f(u) \cdot g(t-u) \cdot du \ .$$

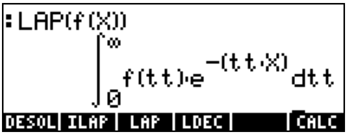
## Laplace transform and inverses in the calculator

The calculator provides the functions LAP and ILAP to calculate the Laplace transform and the inverse Laplace transform, respectively, of a function f(VX), where VX is the CAS default independent variable, which you should set to 'X'. Thus, the calculator returns the transform or inverse transform as a function of X. The functions LAP and ILAP are available under the CALC/DIFF menu. The examples are worked out in the RPN mode, but translating them to ALG mode is straightforward. For these examples, set the CAS mode to Real and Exact.

Example 1 – You can get the definition of the Laplace transform use the following: 'f(X)' ENTER LAP in RPN mode, or LAP(f(X))in ALG mode. The calculator returns the result (RPN, left; ALG, right):



2:
1: $\int_0^\infty f(tt) \cdot e^{-(tt \cdot X)} dtt$



:LAP(f(X))
$\int_0^\infty f(tt) \cdot e^{-(tt \cdot X)} dtt$

Compare these expressions with the one given earlier in the definition of the Laplace transform, i.e.,

$$\mathrm{L}\{f(t)\} = F(s) = \int_0^\infty f(t) \cdot e^{-st} dt,$$

and you will notice that the CAS default variable X in the equation writer screen replaces the variable s in this definition. Therefore, when using the function LAP you get back a function of X, which is the Laplace transform of f(X).

Example 2 – Determine the Laplace transform of f(t) = $e^{2t}\cdot\sin(t)$. Use: ‘EXP(2*X)*SIN(X)’ ENTER LAP The calculator returns the result: 1/(SQ(X-2)+1). Press EVAL to obtain, 1/($X^2$-4X+5).

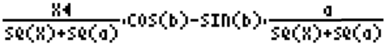
When you translate this result in paper you would write

$$F(s)=\mathrm{L}\{e^{2t}\cdot\sin t\}=\frac{1}{s^{2}-4\cdot s+5}$$

Example 3 – Determine the inverse Laplace transform of F(s) = sin(s). Use: ‘SIN(X)’ ENTER ILAP. The calculator takes a few seconds to return the result: ‘ILAP(SIN(X))’, meaning that there is no closed-form expression f(t), such that f(t) = $\mathrm{L}^{-1}$\{sin(s)\}.

Example 4 – Determine the inverse Laplace transform of F(s) = $1/s^3$. Use: ‘1/X^3’ ENTER ILAP. The calculator returns the result: ‘X^2/2’, which is interpreted as $\mathrm{L}^{-1}\{1/s^3\} = t^2/2$.

Example 5 – Determine the Laplace transform of the function f(t) = cos (a·t+b). Use: ‘COS(a*X+b)’ ENTER LAP . The calculator returns the result:



$$\frac{X}{SQ(X)+SQ(a)}\cdot COS(b)-SIN(b)\cdot\frac{a}{SQ(X)+SQ(a)}$$

Press EVAL to obtain –(a sin(b) – X cos(b))/($X^2$+$a^2$). The transform is interpreted as follows: L \{cos(a·t+b)\} = (s·cos b – a·sin b)/($s^2$+$a^2$).

## Laplace transform theorems

To help you determine the Laplace transform of functions you can use a number of theorems, some of which are listed below. A few examples of the theorem applications are also included.

- Differentiation theorem for the first derivative. Let $f_o$ be the initial condition for f(t), i.e., f(0) = $f_o$, then

$$L\{df/dt\} = s \cdot F(s) - f_o.$$

Example 1 – The velocity of a moving particle v(t) is defined as v(t) = dr/dt, where r = r(t) is the position of the particle. Let $r_o$ = r(0), and R(s) =L{r(t)}, then, the transform of the velocity can be written as V(s) = L{v(t)}=L{dr/dt}= s·R(s)-$r_o$.

- Differentiation theorem for the second derivative. Let $f_o$ = f(0), and $(df/dt)_o = df/dt|_{t=0}$, then $L\{d^2f/dt^2\} = s^2 \cdot F(s) - s \cdot f_o - (df/dt)_o$.

Example 2 – As a follow up to Example 1, the acceleration a(t) is defined as $a(t) = d^2r/dt^2$. If the initial velocity is $v_o = v(0) = dr/dt|_{t=0}$, then the Laplace transform of the acceleration can be written as:

$$A(s) = L\{a(t)\} = L\{d^2r/dt^2\}= s^2 \cdot R(s) - s \cdot r_o - v_o.$$

- Differentiation theorem for the n-th derivative.
  Let $f^{(k)}_o = d^kf/dx^k|_{t=0}$, and $f_o$ = f(0), then

$$L\{d^nf/dt^n\} = s^n \cdot F(s) - s^{n-1} \cdot f_o - \ldots - s \cdot f^{(n-2)}_o - f^{(n-1)}_o.$$

- Linearity theorem. L{af(t)+bg(t)} = a·L{f(t)} + b·L{g(t)}.

- Differentiation theorem for the image function. Let F(s) = L{f(t)}, then $d^nF/ds^n = L\{(-t)^n \cdot f(t)\}$.

Example 3 – Let $f(t) = e^{-at}$, using the calculator with ‘EXP(-a*X)’ ENTER LAP, you get ‘1/(X+a)’, or F(s) = 1/(s+a). The third derivative of this expression can be calculated by using:

‘X’ ENTER ↱ ∂ ‘X’ ENTER ↱ ∂ ‘X’ ENTER ↱ ∂ EVAL

The result is

‘-6/(X^4+4*a*X^3+6*a^2*X^2+4*a^3*X+a^4)’, or

$$d^3F/ds^3 = -6/(s^4+4 \cdot a \cdot s^3+6 \cdot a^2 \cdot s^2+4 \cdot a^3 \cdot s+a^4).$$

Now, use ‘(-X)^3*EXP(-a*X)’ ENTER LAP EVAL. The result is exactly the same.

- Integration theorem. Let F(s) = L{f(t)}, then

$$L\left\{\int_0^t f(u)du\right\}=\frac{1}{s}\cdot F(s).$$

- Convolution theorem. Let F(s) = L{f(t)} and G(s) = L{g(t)}, then

$$L\left\{\int_0^t f(u)g(t-u)du\right\}=L\{(f*g)(t)\}=$$

$$L\{f(t)\}\cdot L\{g(t)\}=F(s)\cdot G(s)$$

Example 4 – Using the convolution theorem, find the Laplace transform of (f*g)(t), if f(t) = sin(t), and g(t) = exp(t). To find F(s) = L{f(t)}, and G(s) = L{g(t)}, use: ‘SIN(X)’ ENTER LAP EVAL. Result, ‘1/(X^2+1)’, i.e., F(s) = $1/(s^2+1)$. Also, ‘EXP(X)’ ENTER LAP. Result, ‘1/(X-1)’, i.e., G(s) = 1/(s-1). Thus, L{(f*g)(t)} = F(s)·G(s) = $1/(s^2+1)\cdot 1/(s-1) = 1/((s-1)(s^2+1)) = 1/(s^3-s^2+s-1)$.

- Shift theorem for a shift to the right. Let F(s) = L{f(t)}, then
$L\{f(t-a)\}=e^{-as}\cdot L\{f(t)\} = e^{-as}\cdot F(s)$.

- Shift theorem for a shift to the left. Let F(s) = L{f(t)}, and a >0, then

$$L\{f(t+a)\}=e^{as}\cdot\left(F(s)-\int_0^a f(t)\cdot e^{-st}\cdot dt\right).$$

- Similarity theorem. Let F(s) = L{f(t)}, and a>0, then L{f(a·t)} = (1/a)·F(s/a).
- Damping theorem. Let F(s) = L{f(t)}, then $L\{e^{-bt}\cdot f(t)\} = F(s+b)$.
- Division theorem. Let F(s) = L{f(t)}, then

$$\mathsf{L}\left\{\frac{f(t)}{t}\right\} = \int_s^\infty F(u)du.$$

- Laplace transform of a periodic function of period T:

$$\mathsf{L}\{f(t)\} = \frac{1}{1-e^{-sT}} \cdot \int_0^T f(t)\cdot e^{-st} \cdot dt.$$

- Limit theorem for the initial value: Let F(s) = L{f(t)}, then

$$f_0 = \lim_{t\to 0} f(t) = \lim_{s\to\infty}[s\cdot F(s)].$$

- Limit theorem for the final value: Let F(s) = L{f(t)}, then

$$f_\infty = \lim_{t\to\infty} f(t) = \lim_{s\to 0}[s\cdot F(s)].$$

## Dirac's delta function and Heaviside's step function

In the analysis of control systems it is customary to utilize a type of functions that represent certain physical occurrences such as the sudden activation of a switch (Heaviside's step function, H(t)) or a sudden, instantaneous, peak in an input to the system (Dirac's delta function, δ(t)). These belong to a class of functions known as generalized or symbolic functions [e.g., see Friedman, B., 1956, Principles and Techniques of Applied Mathematics, Dover Publications Inc., New York (1990 reprint) ].

The formal definition of Dirac's delta function, δ(x), is δ(x) = 0, for x ≠0, and

$$\int_{-\infty}^{\infty} \delta(x)dx = 1.0.$$

Also, if f(x) is a continuous function, then $\int_{-\infty}^{\infty} f(x)\delta(x-x_0)dx = f(x_0).$

An interpretation for the integral above, paraphrased from Friedman (1990), is that the δ-function "picks out" the value of the function f(x) at x = $x_0$. Dirac's delta function is typically represented by an upward arrow at the point x = x0, indicating that the function has a non-zero value only at that particular value of $x_0$.

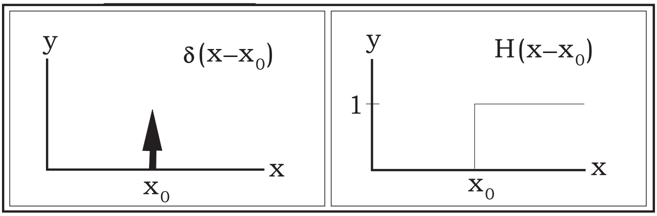
Heaviside's step function, H(x), is defined as

$$H(x)=\begin{cases}1, & x>0\\ 0, & x<0\end{cases}$$

Also, for a continuous function f(x),

$$\int_{-\infty}^{\infty} f(x)H(x-x_0)dx = \int_{x_0}^{\infty} f(x)dx.$$

Dirac's delta function and Heaviside's step function are related by dH/dx = δ(x). The two functions are illustrated in the figure below.



You can prove that L{H(t)} = 1/s,
from which it follows that L{$U_o$·H(t)} = $U_o$/s,
where $U_o$ is a constant. Also, $L^{-1}${1/s}=H(t),
and $L^{-1}${ $U_o$ /s}= $U_o$·H(t).
Also, using the shift theorem for a shift to the right, L{f(t-a)}=$e^{-as}$·L{f(t)} = $e^{-as}$·F(s), we can write L{H(t-k)}=$e^{-ks}$·L{H(t)} = $e^{-ks}$·(1/s) = (1/s)·$e^{-ks}$.

Another important result, known as the <u>second shift theorem for a shift to the right</u>, is that $L^{-1}\{e^{-as}\cdot F(s)\}=f(t-a)\cdot H(t-a)$, with $F(s) = L\{f(t)\}$.

In the calculator the Heaviside step function H(t) is simply referred to as '1'. To check the transform in the calculator use: 1 ENTER LAP. The result is '1/X', i.e., $L\{1\} = 1/s$. Similarly, 'U0' ENTER LAP , produces the result 'U0/X', i.e., $L\{U_0\} = U_0/s$.

You can obtain Dirac's delta function in the calculator by using: 1 ENTER ILAP
The result is 'Delta(X)'.

This result is simply symbolic, i.e., you cannot find a numerical value for, say 'Delta(5)'.

This result can be defined the Laplace transform for Dirac's delta function, because from $L^{-1}\{1.0\}= \delta(t)$, it follows that $L\{\delta(t)\} = 1.0$

Also, using the shift theorem for a shift to the right, $L\{f(t-a)\}=e^{-as}\cdot L\{f(t)\} = e^{-as}\cdot F(s)$, we can write $L\{\delta(t-k)\}=e^{-ks}\cdot L\{\delta(t)\} = e^{-ks}\cdot 1.0 = e^{-ks}$.

## Applications of Laplace transform in the solution of linear ODEs

At the beginning of the section on Laplace transforms we indicated that you could use these transforms to convert a linear ODE in the time domain into an algebraic equation in the image domain. The resulting equation is then solved for a function F(s) through algebraic methods, and the solution to the ODE is found by using the inverse Laplace transform on F(s).

The theorems on derivatives of a function, i.e.,

$$L\{df/dt\} = s\cdot F(s) - f_o,$$

$$L\{d^2f/dt^2\} = s^2\cdot F(s) - s\cdot f_o - (df/dt)_o,$$

and, in general,

$$L\{d^nf/dt^n\} = s^n\cdot F(s) - s^{n-1}\cdot f_o - \ldots - s\cdot f^{(n-2)}_o - f^{(n-1)}{}_o,$$

are particularly useful in transforming an ODE into an algebraic equation.

Example 1 – To solve the first order equation,

$$dh/dt + k\cdot h(t) = a\cdot e^{-t},$$

by using Laplace transforms, we can write:

$$L\{dh/dt + k\cdot h(t)\} = L\{a\cdot e^{-t}\},$$

$$L\{dh/dt\} + k\cdot L\{h(t)\} = a\cdot L\{e^{-t}\}.$$

**Note**: ‘EXP(-X)’ ENTER LAP , produces ‘1/(X+1)’, i.e., $L\{e^{-t}\}=1/(s+1)$.

With $H(s) = L\{h(t)\}$, and $L\{dh/dt\} = s\cdot H(s) - h_o$, where $h_o = h(0)$, the transformed equation is $s\cdot H(s)-h_o+k\cdot H(s) = a/(s+1)$.

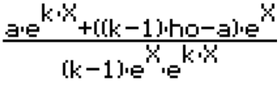
Use the calculator to solve for H(s), by writing:

‘X*H-h0+k*H=a/(X+1)’ ENTER ‘H’ ISOL

The result is ‘H=((X+1)*h0+a)/(X^2+(k+1)*X+k)’.

To find the solution to the ODE, h(t), we need to use the inverse Laplace transform, as follows:

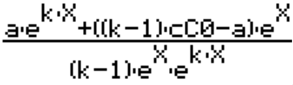
OBJ→ ◄ ◄ EVAL Isolates right-hand side of last expression
ILAP Obtains the inverse Laplace transform



The result is $\frac{a\cdot e^{k\cdot X}+((k-1)\cdot ho-a)\cdot e^{X}}{(k-1)\cdot e^{X}\cdot e^{k\cdot X}}$. Replacing X with t in this expression and simplifying, results in $h(t) = a/(k-1)\cdot e^{-t} +((k-1)\cdot h_o-a)/(k-1)\cdot e^{-kt}$.

Check what the solution to the ODE would be if you use the function LDEC:

‘a*EXP(-X)’ ENTER ‘X+k’ ENTER LDEC EVAL

The result is: 

$$\frac{a\cdot e^{k\cdot X}+((k-1)\cdot cC0-a)\cdot e^{X}}{(k-1)\cdot e^{X}\cdot e^{k\cdot X}}$$

, i.e.,

$$h(t) = a/(k\text{-}1)\cdot e^{-t} +((k\text{-}1)\cdot cC_o\text{-}a)/(k\text{-}1)\cdot e^{-kt}.$$

Thus, cC0 in the results from LDEC represents the initial condition h(0).

**Note**: When using the function LDEC to solve a linear ODE of order n in f(X), the result will be given in terms of n constants cC0, cC1, cC2, …, cC(n-1), representing the initial conditions f(0), f'(0), f"(0), …, $f^{(n\text{-}1)}$ (0).

Example 2 – Use Laplace transforms to solve the second-order linear equation,

$$d^2y/dt^2+2y = \sin 3t.$$

Using Laplace transforms, we can write:

$$L\{d^2y/dt^2+2y\} = L\{\sin 3t\},$$

$$L\{d^2y/dt^2\} + 2\cdot L\{y(t)\} = L\{\sin 3t\}.$$

**Note**: 'SIN(3*X)' ENTER LAP EVAL produces '3/(X^2+9)', i.e., $L\{\sin 3t\}=3/(s^2+9)$.

With Y(s) = L{y(t)}, and $L\{d^2y/dt^2\} = s^2\cdot Y(s) - s\cdot y_o – y_1$, where $y_o$ = h(0) and $y_1$ = h'(0), the transformed equation is

$$s^2\cdot Y(s) – s\cdot y_o – y_1 + 2\cdot Y(s) = 3/(s^2+9).$$

Use the calculator to solve for Y(s), by writing:

'X^2*Y-X*y0-y1+2*Y=3/(X^2+9)' ENTER 'Y' ISOL

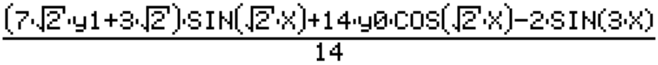
The result is

'Y=((X^2+9)*y1+(y0*X^3+9*y0*X+3))/(X^4+11*X^2+18)'.

To find the solution to the ODE, y(t), we need to use the inverse Laplace transform, as follows:

| | |
|---|---|
| OBJ→ ⬅ ⬅ | Isolates right-hand side of last expression |
| ILAP EVAL | Obtains the inverse Laplace transform |

The result is



$$\frac{(7\cdot\sqrt{2}\cdot y1+3\cdot\sqrt{2})\cdot SIN(\sqrt{2}\cdot X)+14\cdot y0\cdot COS(\sqrt{2}\cdot X)-2\cdot SIN(3\cdot X)}{14}$$

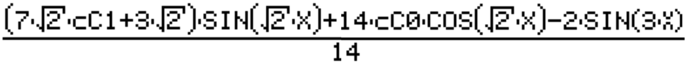
i.e.,

$$y(t) = -(1/7) \sin 3x + y_o \cos \sqrt{}2x + (\sqrt{}2\ (7y_1+3)/14) \sin \sqrt{}2x.$$

Check what the solution to the ODE would be if you use the function LDEC:

‘SIN(3*X)’ ENTER ‘X^2+2’ ENTER LDEC EVAL

The result is:



$$\frac{(7\cdot\sqrt{2}\cdot cC1+3\cdot\sqrt{2})\cdot SIN(\sqrt{2}\cdot X)+14\cdot cC0\cdot COS(\sqrt{2}\cdot X)-2\cdot SIN(3\cdot X)}{14}$$

i.e., the same as before with cC0 = y0 and cC1 = y1.

**Note**: Using the two examples shown here, we can confirm what we indicated earlier, i.e., that function ILAP uses Laplace transforms and inverse transforms to solve linear ODEs given the right-hand side of the equation and the characteristic equation of the corresponding homogeneous ODE.

Example 3 – Consider the equation

$$d^2y/dt^2+y = \delta(t\text{-}3),$$

where $\delta(t)$ is Dirac’s delta function.

Using Laplace transforms, we can write:

$$L\{d^2y/dt^2+y\} = L\{\delta(t\text{-}3)\},$$

$$L\{d^2y/dt^2\} + L\{y(t)\} = L\{\delta(t\text{-}3)\}.$$

With `'Delta(X-3)'` ENTER LAP , the calculator produces EXP(-3*X), i.e., $L\{\delta(t\text{-}3)\} = e^{-3s}$. With $Y(s) = L\{y(t)\}$, and $L\{d^2y/dt^2\} = s^2\cdot Y(s) - s\cdot y_o - y_1$, where $y_o = h(0)$ and $y_1 = h'(0)$, the transformed equation is $s^2\cdot Y(s) - s\cdot y_o - y_1 + Y(s) = e^{-3s}$. Use the calculator to solve for Y(s), by writing:

'X^2*Y-X*y0-y1+Y=EXP(-3*X)' ENTER 'Y' ISOL

The result is 'Y=(X*y0+(y1+EXP(-(3*X))))/(X^2+1)'.

To find the solution to the ODE, y(t), we need to use the inverse Laplace transform, as follows:

OBJ→ ◄ ◄ Isolates right-hand side of last expression
ILAP EVAL Obtains the inverse Laplace transform

The result is 'y1*SIN(X)+y0*COS(X)+SIN(X-3)*Heaviside(X-3)'.

**Notes**:

[1]. An alternative way to obtain the inverse Laplace transform of the expression '(X*y0+(y1+EXP(-(3*X))))/(X^2+1)' is by separating the expression into partial fractions, i.e.,

'y0*X/(X^2+1) + y1/(X^2+1) + EXP(-3*X)/(X^2+1)',

and use the linearity theorem of the inverse Laplace transform

$$L^{-1}\{a\cdot F(s)+b\cdot G(s)\} = a\cdot L^{-1}\{F(s)\} + b\cdot L^{-1}\{G(s)\},$$

to write,

$$L^{-1}\{y_o\cdot s/(s^2+1)+y_1/(s^2+1)) + e^{-3s}/(s^2+1)) \} =$$

$$y_o\cdot L^{-1}\{s/(s^2+1)\}+ y_1\cdot L^{-1}\{1/(s^2+1)\}+ L^{-1}\{e^{-3s}/(s^2+1))\},$$

Then, we use the calculator to obtain the following:

‘X/(X^2+1)’ ENTER ILAP Result, ‘COS(X)’, i.e., L $^{-1}$\{s/(s$^{2}$+1)\}= cos t.
‘1/(X^2+1)’ ENTER ILAP Result, ‘SIN(X)’, i.e., L $^{-1}$\{1/(s$^{2}$+1)\}= sin t.
‘EXP(-3*X)/(X^2+1)’ ENTER ILAP Result, SIN(X-3)*Heaviside(X-3)’.

[2]. The very last result, i.e., the inverse Laplace transform of the expression ‘(EXP(-3*X)/(X^2+1))’, can also be calculated by using the second shifting theorem for a shift to the right)

$$L^{-1}\{e^{-as}\cdot F(s)\}=f(t\text{-}a)\cdot H(t\text{-}a),$$

if we can find an inverse Laplace transform for 1/(s$^{2}$+1). With the calculator, try ‘1/(X^2+1)’ ENTER ILAP. The result is ‘SIN(X)’. Thus, L $^{-1}$\{e$^{-3s}$/(s$^{2}$+1))\} = sin(t-3)·H(t-3),

Check what the solution to the ODE would be if you use the function LDEC:

‘Delta(X-3)’ ENTER ‘X^2+1’ ENTER LDEC EVAL

The result is:

‘SIN(X-3)*Heaviside(X-3) + cC1*SIN(X) + cC0*COS(X)+’.

Please notice that the variable X in this expression actually represents the variable t in the original ODE. Thus, the translation of the solution in paper may be written as:

$$y(t) = Co\cdot\cos t + C_1\cdot\sin t + \sin(t-3)\cdot H(t-3)$$

When comparing this result with the previous result for y(t), we conclude that cC$_{o}$ = y$_{o}$, cC$_{1}$ = y$_{1}$.

### Defining and using Heaviside's step function in the calculator

The previous example provided some experience with the use of Dirac's delta function as input to a system (i.e., in the right-hand side of the ODE describing the system). In this example, we want to use Heaviside's step function, H(t). In the calculator we can define this function as:

'H(X) = IFTE(X>0, 1, 0)' ENTER ↰ DEF

This definition will create the variable H in the calculator's soft menu key.

Example 1 -- To see a plot of H(t-2), for example, use a FUNCTION type of plot (see Chapter 12):

- Press ↰ 2D/3D , simultaneously in RPN mode, to access to the PLOT SETUP window.
- ➢ Change `TYPE` to `FUNCTION`, if needed
- ➢ Change EQ to 'H(X-2)'.
- ➢ Make sure that `Indep` is set to 'X'.
- ➢ Press NXT OK to return to normal calculator display.
- Press ↰ WIN , simultaneously, to access the PLOT window.
- ➢ Change the H-VIEW range to 0 to 20, and the V-VIEW range to -2 to 2.
- ➢ Press ERASE DRAW to plot the function .

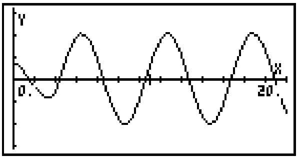
Use of the function H(X) with LDEC, LAP, or ILAP, is not allowed in the calculator. You have to use the main results provided earlier when dealing with the Heaviside step function, i.e., $L\{H(t)\} = 1/s$, $L^{-1}\{1/s\}=H(t)$, $L\{H(t\text{-}k)\}=e^{-ks}\cdot L\{H(t)\} = e^{-ks}\cdot(1/s) = \cdot(1/s)\cdot e^{-ks}$ and $L^{-1}\{e^{-as}\cdot F(s)\}=f(t\text{-}a)\cdot H(t\text{-}a)$.

Example 2 -- The function $H(t\text{-}t_o)$ when multiplied to a function f(t), i.e., $H(t\text{-}t_o)f(t)$, has the effect of switching on the function f(t) at $t = t_o$. For example, the solution obtained in Example 3, above, was $y(t) = y_o \cos t + y_1 \sin t + \sin(t\text{-}3)\cdot H(t\text{-}3)$. Suppose we use the initial conditions $y_o = 0.5$, and $y_1 = -0.25$. Let's plot this function to see what it looks like:

- Press ↰ 2D/3D , simultaneously if in RPN mode, to access to the PLOT SETUP window.

- Change TYPE to FUNCTION, if needed
- Change EQ to ‘0.5*COS(X)-0.25*SIN(X)+SIN(X-3)*H(X-3)’.
- Make sure that Indep is set to ‘X’.
- Press ERASE DRAW to plot the function.
- Press EDIT NXT LABEL to see the plot.

The resulting graph will look like this:



Notice that the signal starts with a relatively small amplitude, but suddenly, at t=3, it switches to an oscillatory signal with a larger amplitude. The difference between the behavior of the signal before and after t = 3 is the “switching on” of the particular solution $y_p(t) = \sin(t-3)\cdot H(t-3)$. The behavior of the signal before t = 3 represents the contribution of the homogeneous solution, $y_h(t) = y_o \cos t + y_1 \sin t$.

The solution of an equation with a driving signal given by a Heaviside step function is shown below.

<u>Example 3</u> – Determine the solution to the equation, $d^2y/dt^2+y = H(t-3)$, where H(t) is Heaviside’s step function. Using Laplace transforms, we can write: $L\{d^2y/dt^2+y\} = L\{H(t-3)\}$, $L\{d^2y/dt^2\} + L\{y(t)\} = L\{H(t-3)\}$. The last term in this expression is: $L\{H(t-3)\} = (1/s)\cdot e^{-3s}$. With $Y(s) = L\{y(t)\}$, and $L\{d^2y/dt^2\} = s^2\cdot Y(s) - s\cdot y_o - y_1$, where $y_o = h(0)$ and $y_1 = h'(0)$, the transformed equation is $s^2\cdot Y(s) - s\cdot y_o - y_1 + Y(s) = (1/s)\cdot e^{-3s}$. Change CAS mode to Exact, if necessary. Use the calculator to solve for Y(s), by writing:

‘X^2*Y-X*y0-y1+Y=(1/X)*EXP(-3*X)’ ENTER ‘Y’ ISOL

The result is ‘Y=(X^2*y0+X*y1+EXP(-3*X))/(X^3+X)’.

To find the solution to the ODE, y(t), we need to use the inverse Laplace transform, as follows:

OBJ→ ◄ ◄ Isolates right-hand side of last expression
ILAP Obtains the inverse Laplace transform

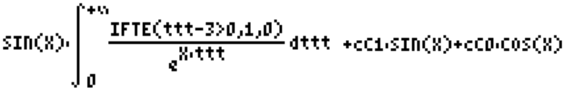
The result is ‘y1*SIN(X-1)+y0*COS(X-1)-(COS(X-3)-1)*Heaviside(X-3)’.

Thus, we write as the solution: y(t) = $y_o$ cos t + $y_1$ sin t + H(t-3)·(1+sin(t-3)).

Check what the solution to the ODE would be if you use the function LDEC:

‘H(X-3)’ ENTER [ENTER] ‘X^2+1’ ENTER LDEC

The result is:



$$\text{SIN}(X)\cdot\int_{0}^{+\infty}\frac{\text{IFTE}(ttt-3>0,1,0)}{e^{X\cdot ttt}}\,dttt\ +cC1\cdot\text{SIN}(X)+cC0\cdot\text{COS}(X)$$

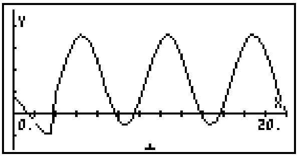
Please notice that the variable X in this expression actually represents the variable t in the original ODE, and that the variable *ttt* in this expression is a dummy variable. Thus, the translation of the solution in paper may be written as:

$$y(t) = Co\cdot\cos t + C_1\cdot\sin t + \sin t\cdot\int_0^\infty H(u-3)\cdot e^{-ut}\cdot du.$$

Example 4 – Plot the solution to Example 3 using the same values of $y_o$ and $y_1$ used in the plot of Example 1, above. We now plot the function

y(t) = 0.5 cos t –0.25 sin t + (1+sin(t-3))·H(t-3).

In the range 0 < t < 20, and changing the vertical range to (-1,3), the graph should look like this:



Again, there is a new component to the motion switched at t=3, namely, the particular solution $y_p(t) = [1+\sin(t\text{-}3)]\cdot H(t\text{-}3)$, which changes the nature of the solution for t>3.

The Heaviside step function can be combined with a constant function and with linear functions to generate square, triangular, and saw tooth finite pulses, as follows:

- Square pulse of size $U_o$ in the interval $a < t < b$:

$$f(t) = Uo[H(t\text{-}a)\text{-}H(t\text{-}b)].$$

- Triangular pulse with a maximum value Uo, increasing from $a < t < b$, decreasing from $b < t < c$:

$$f(t) = U_o\cdot\ ((t\text{-}a)/(b\text{-}a)\cdot[H(t\text{-}a)\text{-}H(t\text{-}b)]+(1\text{-}(t\text{-}b)/(b\text{-}c))[H(t\text{-}b)\text{-}H(t\text{-}c)]).$$

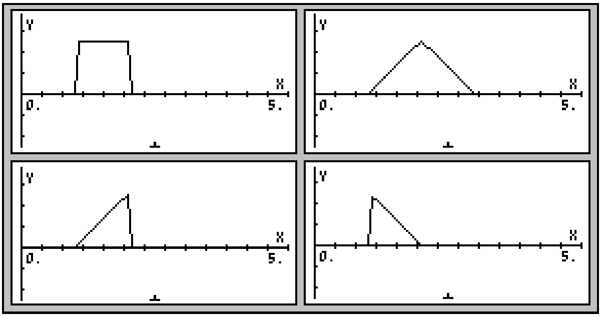
- Saw tooth pulse increasing to a maximum value Uo for $a < t < b$, dropping suddenly down to zero at $t = b$:

$$f(t) = U_o\cdot\ (t\text{-}a)/(b\text{-}a)\cdot[H(t\text{-}a)\text{-}H(t\text{-}b)].$$

- Saw tooth pulse increasing suddenly to a maximum of Uo at $t = a$, then decreasing linearly to zero for $a < t < b$:

$$f(t) = U_o\cdot[1\text{-}(t\text{-}a)/(b\text{-}1)]\cdot[H(t\text{-}a)\text{-}H(t\text{-}b)].$$

Examples of the plots generated by these functions, for Uo = 1, a = 2, b = 3, c = 4, horizontal range = (0,5), and vertical range = (-1, 1.5), are shown in the figures below:



## Fourier series

Fourier series are series involving sine and cosine functions typically used to expand periodic functions. A function f(x) is said to be *periodic*, of period T, if f(x+T) = f(t). For example, because sin(x+2π) = sin x, and cos(x+2π) = cos x, the functions *sin* and *cos* are 2π-periodic functions. If two functions f(x) and g(x) are periodic of period T, then their linear combination h(x) = a·f(x) + b·g(x), is also periodic of period T. A T-periodic function f(t) can be expanded into a series of sine and cosine functions known as a Fourier series given by

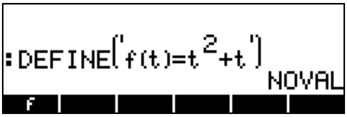
$$f(t) = a_0 + \sum_{n=1}^{\infty}\left( a_n \cdot \cos\frac{2n\pi}{T}t + b_n \cdot \sin\frac{2n\pi}{T}t \right)$$

where the coefficients $a_n$ and $b_n$ are given by

$$a_0 = \frac{1}{T}\int_{-T/2}^{T/2} f(t)\cdot dt, \quad a_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cdot\cos\frac{2n\pi}{T}t\cdot dt,$$

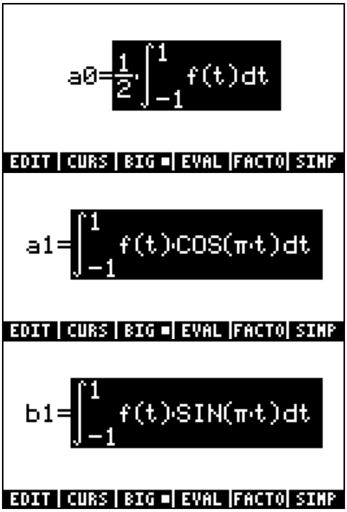
$$b_n = \int_{-T/2}^{T/2} f(t)\cdot\sin\frac{2n\pi}{T}t\cdot dt.$$

The following exercises are in ALG mode, with CAS mode set to Exact. (When you produce a graph, the CAS mode will be reset to Approx. Make sure to set it back to Exact after producing the graph.) Suppose, for example, that the function f(t) = $t^2$+t is periodic with period T = 2. To determine the coefficients $a_0$, $a_1$, and $b_1$ for the corresponding Fourier series, we proceed as follows: First, define function f(t) = $t^2$+t :



```
:DEFINE('f(t)=t^2+t')
                  NOVAL
```

Next, we'll use the Equation Writer to calculate the coefficients:

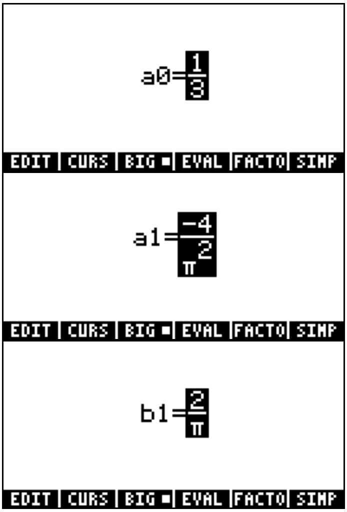


$$a0=\frac{1}{2}\cdot\int_{-1}^{1}f(t)dt$$

$$a1=\int_{-1}^{1}f(t)\cdot COS(\pi\cdot t)dt$$

$$b1=\int_{-1}^{1}f(t)\cdot SIN(\pi\cdot t)dt$$

EDIT | CURS | BIG ■ | EVAL | FACTO | SIMP
EDIT | CURS | BIG ■ | EVAL | FACTO | SIMP
EDIT | CURS | BIG ■ | EVAL | FACTO | SIMP



$$a0=\frac{1}{3}$$

$$a1=\frac{-4}{\pi^2}$$

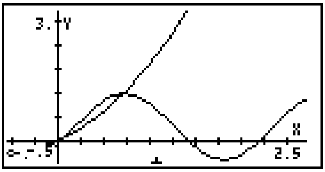
$$b1=\frac{2}{\pi}$$

EDIT | CURS | BIG ■ | EVAL | FACTO | SIMP
EDIT | CURS | BIG ■ | EVAL | FACTO | SIMP
EDIT | CURS | BIG ■ | EVAL | FACTO | SIMP

Thus, the first three terms of the function are:

$$f(t) \approx 1/3 - (4/\pi^2)\cdot\cos(\pi\cdot t)+(2/\pi)\cdot\sin(\pi\cdot t).$$

A graphical comparison of the original function with the Fourier expansion using these three terms shows that the fitting is acceptable for t < 1, or thereabouts. But, then, again, we stipulated that T/2 = 1. Therefore, the fitting is valid only between –1 < t < 1.



## Function FOURIER

An alternative way to define a Fourier series is by using complex numbers as follows:

$$f(t)=\sum_{n=-\infty}^{+\infty}c_n\cdot\exp(\frac{2in\pi t}{T}),$$

where

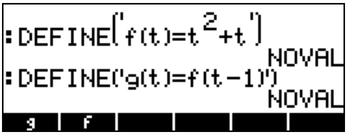
$$c_n = \frac{1}{T}\int_0^T f(t)\cdot \exp(-\frac{2\cdot i\cdot n\cdot \pi}{T}\cdot t)\cdot dt, \quad n = -\infty,...,-2,-1,0,1,2,...\infty.$$

Function FOURIER provides the coefficient $c_n$ of the complex-form of the Fourier series given the function f(t) and the value of n. The function FOURIER requires you to store the value of the period (T) of a T-periodic function into the CAS variable PERIOD before calling the function. The function FOURIER is available in the DERIV sub-menu within the CALC menu (← CALC ).

## Fourier series for a quadratic function

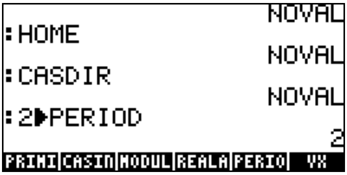
Determine the coefficients $c_0$, $c_1$, and $c_2$ for the function f(t) = $t^2$+t, with period T = 2. (Note: Because the integral used by function FOURIER is calculated in the interval [0,T], while the one defined earlier was calculated in the interval [-T/2,T/2], we need to shift the function in the t-axis, by subtracting T/2 from t, i.e., we will use g(t) = f(t-1) = $(t-1)^2$+(t-1).)

Using the calculator in ALG mode, first we define functions f(t) and g(t):



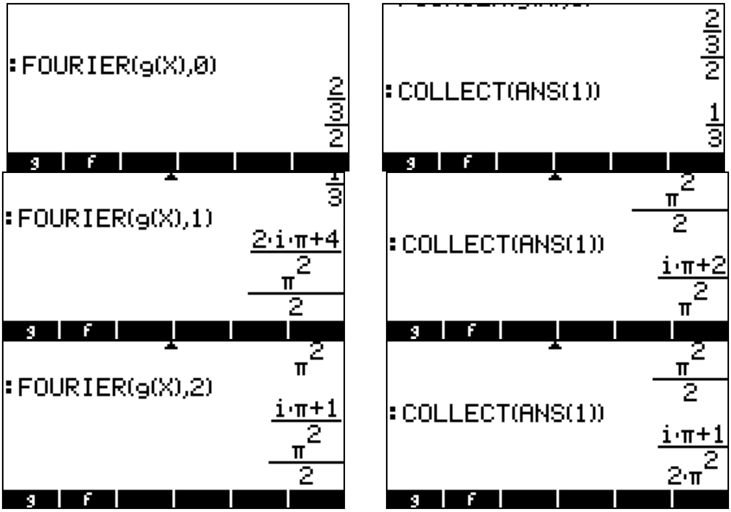
```
:DEFINE('f(t)=t^2+t')
                    NOVAL
:DEFINE('g(t)=f(t-1)')
                    NOVAL
```

Next, we move to the CASDIR sub-directory under HOME to change the value of variable PERIOD, e.g., ← (hold) UPDIR ENTER VAR CASDI ENTER 2 STO▶ PERIO ENTER



```
                    NOVAL
:HOME
                    NOVAL
:CASDIR
                    NOVAL
:2▶PERIOD
                        2
```

Return to the sub-directory where you defined functions f and g, and calculate the coefficients (Accept change to Complex mode when requested):



:FOURIER(g(X),0)

$$\frac{\frac{2}{3}}{2}$$

:COLLECT(ANS(1))

$$\frac{1}{3}$$

:FOURIER(g(X),1)

$$\frac{2\cdot i\cdot \pi+4}{\frac{\pi^2}{2}}$$

:COLLECT(ANS(1))

$$\frac{i\cdot \pi+2}{\pi^2}$$

:FOURIER(g(X),2)

$$\frac{i\cdot \pi+1}{\frac{\pi^2}{2}}$$

:COLLECT(ANS(1))

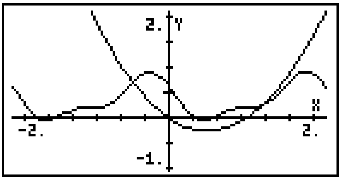
$$\frac{i\cdot \pi+1}{2\cdot \pi^2}$$

Thus, $c_0 = 1/3$, $c_1 = (\pi\cdot i+2)/\pi^2$, $c_2 = (\pi\cdot i+1)/(2\pi^2)$.

The Fourier series with three elements will be written as

$$g(t) \approx \text{Re}[(1/3) + (\pi\cdot i+2)/\pi^2\cdot\exp(i\cdot\pi\cdot t)+ (\pi\cdot i+1)/(2\pi^2)\cdot\exp(2\cdot i\cdot\pi\cdot t)].$$

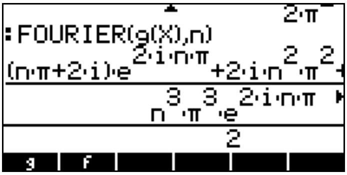
A plot of the shifted function g(t) and the Fourier series fitting follows:



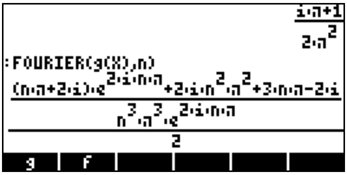
The fitting is somewhat acceptable for 0<t<2, although not as good as in the previous example.

### A general expression for $c_n$

The function FOURIER can provide a general expression for the coefficient $c_n$ of the complex Fourier series expansion. For example, using the same function g(t) as before, the general term $c_n$ is given by (figures show normal font and small font displays):



```
:FOURIER(g(X),n)
(n·π+2·i)·e^(2·i·n·π)+2·i·n^2·π^2+
        n^3·π^3·e^(2·i·n·π)
        2
```



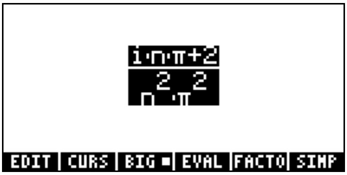
```
                                   i·π+1
                                   2·π^2
:FOURIER(g(X),n)
(n·π+2·i)·e^(2·i·n·π)+2·i·n^2·π^2+3·n·π-2·i
        n^3·π^3·e^(2·i·n·π)
        2
```

The general expression turns out to be, after simplifying the previous result,

$$c_n = \frac{(n\pi + 2i)\cdot e^{2in\pi} + 2i^2 n^2 \pi^2 + 3n\pi - 2i}{2n^3\pi^3 \cdot e^{2in\pi}}$$

We can simplify this expression even further by using Euler's formula for complex numbers, namely, $e^{2in\pi} = \cos(2n\pi) + i\cdot\sin(2n\pi) = 1 + i\cdot 0 = 1$, since $\cos(2n\pi) = 1$, and $\sin(2n\pi) = 0$, for n integer.

Using the calculator you can simplify the expression in the equation writer (↱ EQW) by replacing $e^{2in\pi} = 1$. The figure shows the expression after simplification:



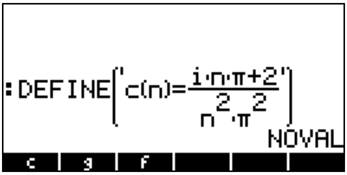
```
i·n·π+2
n^2·π^2
```

The result is $c_n = (i\cdot n\cdot\pi+2)/(n^2\cdot\pi^2)$.

### Putting together the complex Fourier series

Having determined the general expression for $c_n$, we can put together a finite complex Fourier series by using the summation function (Σ) in the calculator as follows:

- First, define a function c(n) representing the general term $c_n$ in the complex Fourier series.



$$:\mathrm{DEFINE}\left('c(n)=\frac{i\cdot n\cdot \pi+2}{n^2\cdot \pi^2}'\right)$$

NOVAL

- Next, define the finite complex Fourier series, F(X,k), where X is the independent variable and k determines the number of terms to be used. Ideally we would like to write this finite complex Fourier series as

$$F(X,k)=\sum_{n=-k}^{k} c(n)\cdot \exp(\frac{2\cdot i\cdot \pi\cdot n}{T}\cdot X)$$

However, because the function c(n) is not defined for n = 0, we will be better advised to re-write the expression as

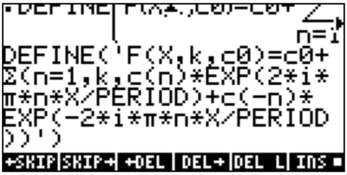
$$F(X,k,c0)=c0+$$

$$\sum_{n=1}^{k}[c(n)\cdot \exp(\frac{2\cdot i\cdot \pi\cdot n}{T}\cdot X)+c(-n)\cdot \exp(-\frac{2\cdot i\cdot \pi\cdot n}{T}\cdot X)],$$

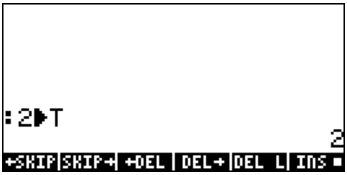
Or, in the calculator entry line as:

DEFINE('F(X,k,c0) = c0+Σ(n=1,k,c(n)*EXP(2*i*π*n*X/T)+
c(-n)*EXP(-(2*i*π*n*X/T))'),

where T is the period, T = 2. The following screen shots show the definition of function F and the storing of T = 2:

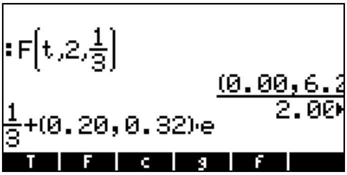


```
DEFINE('F(X,k,c0)=c0+
Σ(n=1,k,c(n)*EXP(2*i*
π*n*X/PERIOD)+c(-n)*
EXP(-2*i*π*n*X/PERIOD
))')
```



```
:2▶T
                    2
```

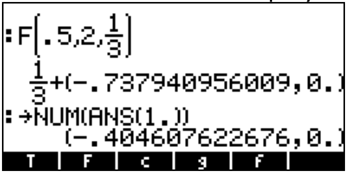
The function F can be used to generate the expression for the complex Fourier series for a finite value of k. For example, for k = 2, $c_0$ = 1/3,and using t as the independent variable, we can evaluate F(t,2,1/3) to get:



```
:F[t,2,1/3]
                 (0.00,6.2
                   2.00▸
1/3+(0.20,0.32)·e
```

This result shows only the first term (c0) and part of the first exponential term in the series. The decimal display format was changed to Fix with 2 decimals to be able to show some of the coefficients in the expansion and in the exponent. As expected, the coefficients are complex numbers.

The function F, thus defined, is fine for obtaining values of the finite Fourier series. For example, a single value of the series, e.g., F(0.5,2,1/3), can be obtained by using (CAS modes set to Exact, step-by-step, and Complex):



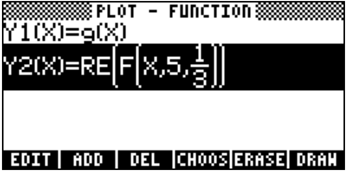
```
:F[.5,2,1/3]
  1/3+(-.737940956009,0.)
:→NUM(ANS(1.))
     (-.404607622676,0.)
```

Accept change to `Approx` mode if requested. The result is the value –0.40467…. The actual value of the function g(0.5) is g(0.5) = -0.25. The following calculations show how well the Fourier series approximates this value as the number of components in the series, given by k, increases:

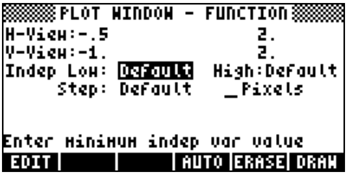
F (0.5, 1, 1/3) = (-0.303286439037,0.)
F (0.5, 2, 1/3) = (-0.404607622676,0.)
F (0.5, 3, 1/3) = (-0.192401031886,0.)

F (0.5, 4, 1/3) = (-0.167070735979,0.)
F (0.5, 5, 1/3) = (-0.294394690453,0.)
F (0.5, 6, 1/3) = (-0.305652599743,0.)

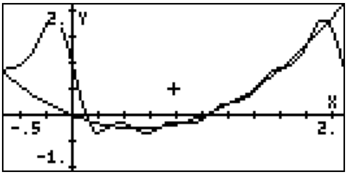
To compare the results from the series with those of the original function, load these functions into the PLOT – FUNCTION input form (↰ Y= , simultaneously if using RPN mode):


PLOT - FUNCTION
Y1(X)=g(X)
Y2(X)=RE(F(X,5,$\frac{1}{3}$))

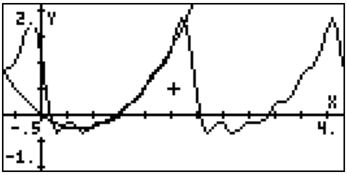
Change the limits of the Plot Window (↰ WIN ) as follows:


PLOT WINDOW - FUNCTION
H-View: -.5    2.
V-View: -1.    2.
Indep Low: Default    High: Default
Step: Default    _Pixels
Enter minimum indep var value

Press the soft-menu keys ERASE DRAW to produce the plot:



Notice that the series, with 5 terms, "hugs" the graph of the function very closely in the interval 0 to 2 (i.e., through the period T = 2). You can also notice a periodicity in the graph of the series. This periodicity is easy to visualize by expanding the horizontal range of the plot to (-0.5,4):



## Fourier series for a triangular wave

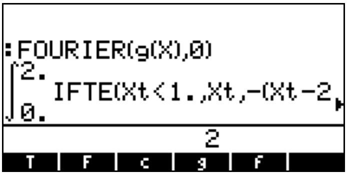
Consider the function

$$g(x) = \begin{cases} x, & if\ 0 < x < 1 \\ 2 - x, & if\ 1 < x < 2 \end{cases}$$

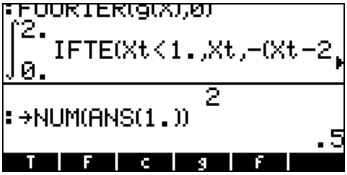
which we assume to be periodic with period T = 2. This function can be defined in the calculator, in ALG mode, by the expression

DEFINE('g(X) = IFTE(X<1,X,2-X)')

If you started this example after finishing example 1 you already have a value of 2 stored in CAS variable PERIOD. If you are not sure, check the value of this variable, and store a 2 in it if needed. The coefficient $c_0$ for the Fourier series is calculated as follows:

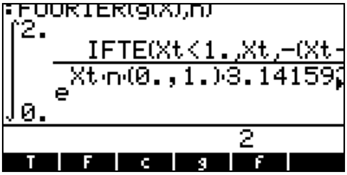


```
:FOURIER(g(X),0)
 2.
∫   IFTE(Xt<1.,Xt,-(Xt-2
 0.
          2
```



```
:FOURIER(g(X),0)
 2.
∫   IFTE(Xt<1.,Xt,-(Xt-2,
 0.
                 2
:→NUM(ANS(1.))
                         .5
```

The calculator will request a change to Approx mode because of the integration of the function IFTE() included in the integrand. Accepting, the change to Approx produces $c_0 = 0.5$. If we now want to obtain a generic expression for the coefficient $c_n$ use:



```
:FOURIER(g(X),n)
```

$$\int_{0.}^{2.}\frac{\mathrm{IFTE}(Xt<1.,Xt,-(Xt-}{e^{Xt\cdot n\cdot(0.,1.)\cdot 3.14159}}$$

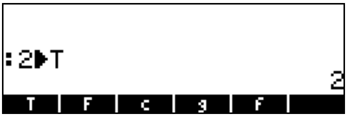
```
2
```

The calculator returns an integral that cannot be evaluated numerically because it depends on the parameter n. The coefficient can still be calculated by typing its definition in the calculator, i.e.,

$$\frac{1}{2}\cdot\int_0^1 X\cdot EXP\left(-\frac{i\cdot 2\cdot n\cdot\pi\cdot X}{T}\right)\cdot dX +$$

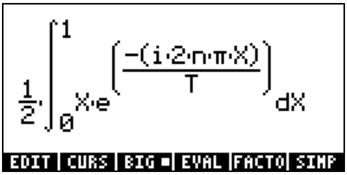
$$\frac{1}{2}\cdot\int_1^2 (2-X)\cdot EXP\left(-\frac{i\cdot 2\cdot n\cdot\pi\cdot X}{T}\right)\cdot dX$$

where T = 2 is the period. The value of T can be stored using:

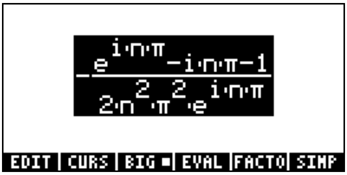


```
:2▶T
                    2
```

Typing the first integral above in the Equation Writer, selecting the entire expression, and using EVAL, will produce the following:

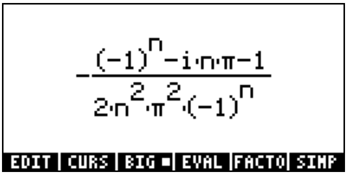


$$\frac{1}{2}\cdot\int_0^1 X\cdot e^{\left(\frac{-(i\cdot 2\cdot n\cdot\pi\cdot X)}{T}\right)}dX$$

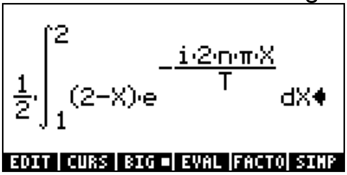


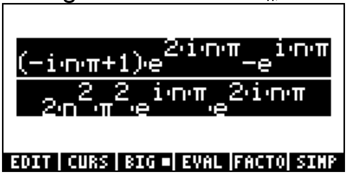
$$-\frac{e^{i\cdot n\cdot\pi}-i\cdot n\cdot\pi-1}{2\cdot n^{2}\cdot\pi^{2}\cdot e^{i\cdot n\cdot\pi}}$$

Recall the $e^{in\pi} = \cos(n\pi) + i\cdot\sin(n\pi) = (-1)^n$ . Performing this substitution in the result above we have:

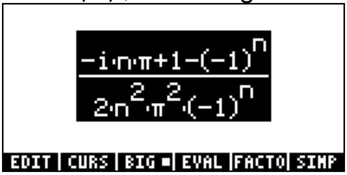

$$-\frac{(-1)^n-i\cdot n\cdot\pi-1}{2\cdot n^2\cdot\pi^2\cdot(-1)^n}$$

Press ENTER ENTER to copy this result to the screen. Then, reactivate the Equation Writer to calculate the second integral defining the coefficient $c_n$, namely,

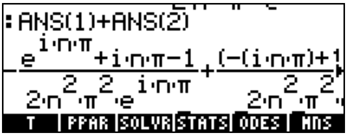

$$\frac{1}{2}\cdot\int_1^2 (2-X)\cdot e^{-\frac{i\cdot 2\cdot n\cdot\pi\cdot X}{T}}\,dX$$


$$\frac{(-i\cdot n\cdot\pi+1)\cdot e^{2\cdot i\cdot n\cdot\pi}-e^{i\cdot n\cdot\pi}}{2\cdot n^2\cdot\pi^2\cdot e^{i\cdot n\cdot\pi}\cdot e^{2\cdot i\cdot n\cdot\pi}}$$

Once again, replacing $e^{in\pi} = (-1)^n$, and using $e^{2in\pi} = 1$, we get:

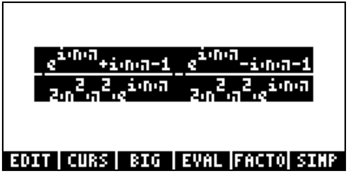

$$\frac{-i\cdot n\cdot\pi+1-(-1)^n}{2\cdot n^2\cdot\pi^2\cdot(-1)^n}$$

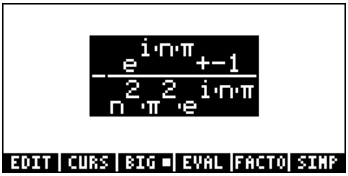
Press ENTER ENTER to copy this second result to the screen. Now, add ANS(1) and ANS(2) to get the full expression for $c_n$:


:ANS(1)+ANS(2)

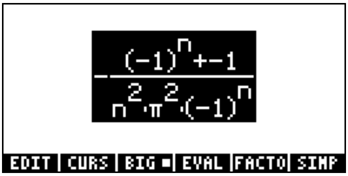
$$-\frac{e^{i\cdot n\cdot\pi}+i\cdot n\cdot\pi-1}{2\cdot n^2\cdot\pi^2\cdot e^{i\cdot n\cdot\pi}}+\frac{(-(i\cdot n\cdot\pi)+1}{2\cdot n^2\cdot\pi^2\cdot}$$

Pressing ▼ will place this result in the Equation Writer, where we can simplify (SIMP) it to read:


$$-\frac{e^{i\cdot n\cdot\pi}+i\cdot n\cdot\pi-1}{2\cdot n^2\cdot\pi^2\cdot e^{i\cdot n\cdot\pi}}-\frac{e^{i\cdot n\cdot\pi}-i\cdot n\cdot\pi-1}{2\cdot n^2\cdot\pi^2\cdot e^{i\cdot n\cdot\pi}}$$


$$-\frac{e^{i\cdot n\cdot\pi}+-1}{n^2\cdot\pi^2\cdot e^{i\cdot n\cdot\pi}}$$

Once again, replacing $e^{in\pi}$ = $(-1)^n$, results in

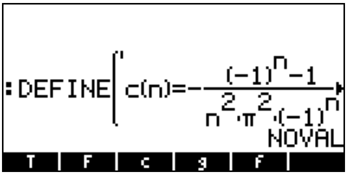


$$-\frac{(-1)^n+-1}{n^2\cdot\pi^2\cdot(-1)^n}$$

This result is used to define the function c(n) as follows:

DEFINE('c(n) = - (((-1)^n-1)/(n^2*π^2*(-1)^n)')

i.e.,



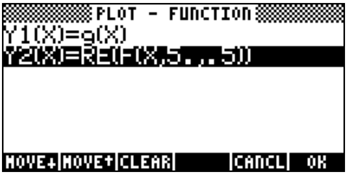
$$:\mathrm{DEFINE}\left('c(n)=-\frac{(-1)^n-1}{n^2\cdot\pi^2\cdot(-1)^n}\right.$$

NOVAL

Next, we define function F(X,k,c0) to calculate the Fourier series (if you completed example 1, you already have this function stored):

DEFINE('F(X,k,c0) = c0+Σ(n=1,k,c(n)*EXP(2*i*π*n*X/T)+
c(-n)*EXP(-(2*i*π*n*X/T))'),

To compare the original function and the Fourier series we can produce the simultaneous plot of both functions. The details are similar to those of example 1, except that here we use a horizontal range of 0 to 2 and a vertical range from 0 to 1, and adjust the equations to plot as shown here:

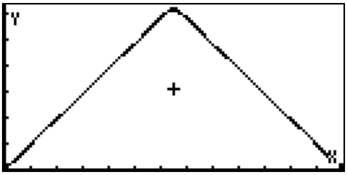


PLOT - FUNCTION

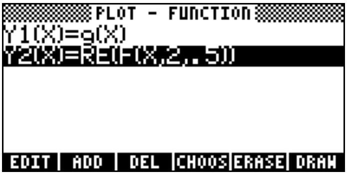
Y1(X)=g(X)

Y2(X)=RE(F(X,5.,.5))

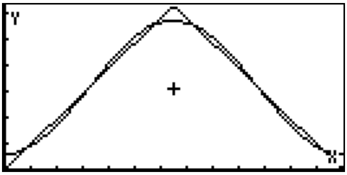
The resulting graph is shown below for k = 5 (the number of elements in the series is 2k+1, i.e., 11, in this case):



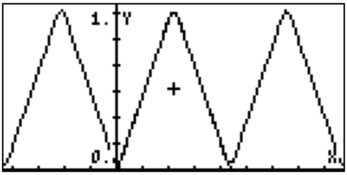
From the plot it is very difficult to distinguish the original function from the Fourier series approximation. Using k = 2, or 5 terms in the series, shows not so good a fitting:



```
PLOT - FUNCTION
Y1(X)=g(X)
Y2(X)=RE(F(X,2,.5))
```



The Fourier series can be used to generate a periodic triangular wave (or saw tooth wave) by changing the horizontal axis range, for example, from –2 to 4. The graph shown below uses k = 5:



## Fourier series for a square wave

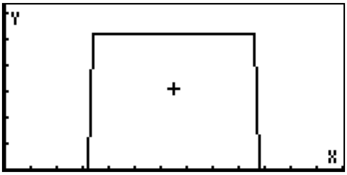
A square wave can be generated by using the function

$$g(x) = \begin{cases} 0, & if\ 0 < x < 1 \\ 1, & if\ 1 < x < 3 \\ 0, & if\ 3 < x < 4 \end{cases}$$

In this case, the period T, is 4. Make sure to change the value of variable T to 4 (use: 4 STO▶ T ENTER). Function g(X) can be defined in the calculator by using

DEFINE('g(X) = IFTE((X>1) AND (X<3),1,0)')

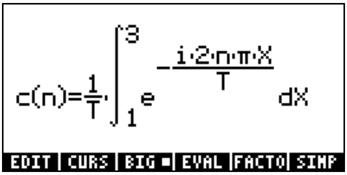
The function plotted as follows (horizontal range: 0 to 4, vertical range:0 to 1.2 ):



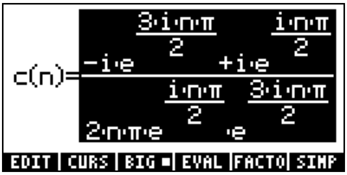
Using a procedure similar to that of the triangular shape in example 2, above, you can find that

$$c_0 = \frac{1}{T}\cdot\left(\int_1^3 1\cdot dX\right) = 0.5\,,$$

and

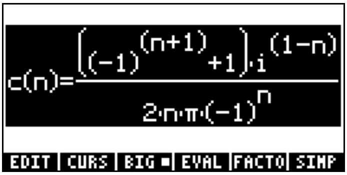


$$c(n)=\frac{1}{T}\cdot\int_1^3 e^{-\frac{i\cdot 2\cdot n\cdot \pi\cdot X}{T}}\,dX$$

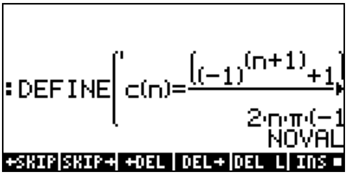


$$c(n)=\frac{-i\cdot e^{\frac{3\cdot i\cdot n\cdot \pi}{2}}+i\cdot e^{\frac{i\cdot n\cdot \pi}{2}}}{2\cdot n\cdot \pi\cdot e^{\frac{i\cdot n\cdot \pi}{2}}\cdot e^{\frac{3\cdot i\cdot n\cdot \pi}{2}}}$$

We can simplify this expression by using $e^{in\pi/2} = i^n$ and $e^{3in\pi/2} = (-i)^n$ to get:



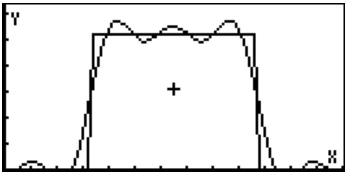
$c(n)=\frac{\left((-1)^{(n+1)}+1\right)\cdot i^{(1-n)}}{2\cdot n\cdot \pi\cdot (-1)^{n}}$



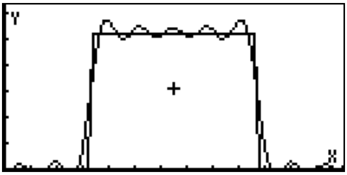
:DEFINE$\left(' c(n)=\frac{\left((-1)^{(n+1)}+1\right.}{2\cdot n\cdot \pi\cdot(-1}\right.$

NOVAL

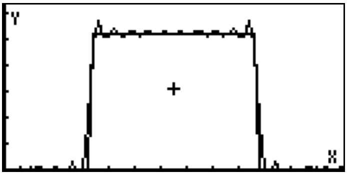
The simplification of the right-hand side of c(n), above, is easier done on paper (i.e., by hand). Then, retype the expression for c(n) as shown in the figure to the left above, to define function c(n). The Fourier series is calculated with F(X,k,c0), as in examples 1 and 2 above, with c0 = 0.5. For example, for k = 5, i.e., with 11 components, the approximation is shown below:



A better approximation is obtained by using k = 10, i.e.,



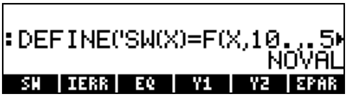
For k = 20, the fitting is even better, but it takes longer to produce the graph:



## Fourier series applications in differential equations

Suppose we want to use the periodic square wave defined in the previous example as the excitation of an undamped spring-mass system whose homogeneous equation is: $d^2y/dX^2 + 0.25y = 0$.

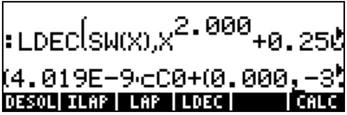
We can generate the excitation force by obtaining an approximation with k =10 out of the Fourier series by using SW(X) = F(X,10,0.5):



```
:DEFINE('SW(X)=F(X,10,.5▸
                    NOVAL
```

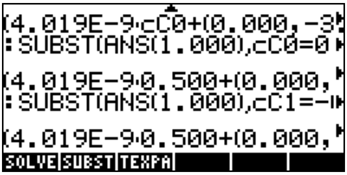
We can use this result as the first input to the function LDEC when used to obtain a solution to the system $d^2y/dX^2 + 0.25y = SW(X)$, where SW(X) stands for Square Wave function of X. The second input item will be the characteristic equation corresponding to the homogeneous ODE shown above, i.e., 'X^2+0.25' .

With these two inputs, function LDEC produces the following result (decimal format changed to Fix with 3 decimals).



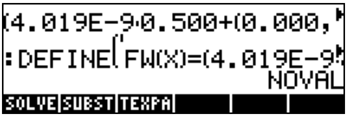
```
:LDEC(SW(X),X^2.000+0.25
(4.019E-9·cC0+(0.000,-3
```

Pressing ▼ allows you to see the entire expression in the Equation writer. Exploring the equation in the Equation Writer reveals the existence of two constants of integration, cC0 and cC1. These values would be calculated using initial conditions. Suppose that we use the values cC0 = 0.5 and cC1 = -0.5, we can replace those values in the solution above by using function SUBST (see Chapter 5). For this case, use SUBST(ANS(1),cC0=0.5) ENTER, followed by SUBST(ANS(1),cC1=-0.5) ENTER. Back into normal calculator display we can use:



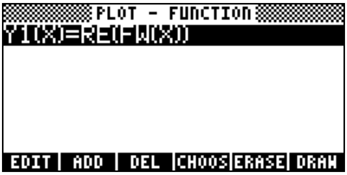
```
(4.019E-9·cC0+(0.000,-3
:SUBST(ANS(1.000),cC0=0
(4.019E-9·0.500+(0.000,
:SUBST(ANS(1.000),cC1=-
(4.019E-9·0.500+(0.000,
```

The latter result can be defined as a function, FW(X), as follows (cutting and pasting the last result into the command):

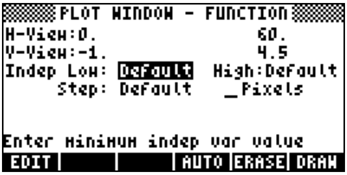


```
(4.019E-9·0.500+(0.000,
:DEFINE('FW(X)=(4.019E-9
                   NOVAL
```

We can now plot the real part of this function. Change the decimal mode to Standard, and use the following:

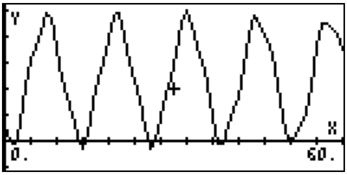


```
PLOT - FUNCTION
Y1(X)=RE(FW(X))
```



```
PLOT WINDOW - FUNCTION
H-View:0.              60.
V-View:-1.             4.5
Indep Low: Default  High:Default
     Step: Default   _Pixels

Enter minimum indep var value
```

The solution is shown below:



## Fourier Transforms

Before presenting the concept of Fourier transforms, we'll discuss the general definition of an integral transform. In general, an <u>integral transform</u> is a transformation that relates a function f(t) to a new function F(s) by an integration of the form $F(s)=\int_a^b \kappa(s,t)\cdot f(t)\cdot dt.$ The function κ(s,t) is known as the kernel of the transformation.

The use of an integral transform allows us to resolve a function into a given spectrum of components. To understand the concept of a spectrum, consider the Fourier series

$$f(t)=a_0+\sum_{n=1}^{\infty}\left(a_n\cdot\cos\omega_n x+b_n\cdot\sin\omega_n x\right),$$

representing a periodic function with a period T. This Fourier series can be re-written as $f(x)=a_0+\sum_{n=1}^{\infty}A_n\cdot\cos(\varpi_n x+\phi_n),$ where

$$A_n=\sqrt{a_n^2+b_n^2},\quad \phi_n=\tan^{-1}\left(\frac{b_n}{a_n}\right),$$

for n =1,2, …

The amplitudes $A_n$ will be referred to as the spectrum of the function and will be a measure of the magnitude of the component of f(x) with frequency $f_n$ = n/T. The basic or fundamental frequency in the Fourier series is $f_0$ = 1/T, thus, all other frequencies are multiples of this basic frequency, i.e., $f_n = n\cdot f_0$. Also, we can define an angular frequency, $\omega_n = 2n\pi/T = 2\pi\cdot f_n = 2\pi\cdot n\cdot f_0 = n\cdot\omega_0$, where $\omega_0$ is the basic or fundamental angular frequency of the Fourier series.

Using the angular frequency notation, the Fourier series expansion is written as

$$f(x)=a_0+\sum_{n=1}^{\infty}A_n\cdot\cos(\omega_n x+\phi_n).$$

$$=a_0+\sum_{n=1}^{\infty}\left(a_n\cdot\cos\omega_n x+b_n\cdot\sin\omega_n x\right)$$

A plot of the values $A_n$ vs. $\omega_n$ is the typical representation of a discrete spectrum for a function. The discrete spectrum will show that the function has components at angular frequencies $\omega_n$ which are integer multiples of the fundamental angular frequency $\omega_0$.

Suppose that we are faced with the need to expand a non-periodic function into sine and cosine components. A non-periodic function can be thought of as having an infinitely large period. Thus, for a very large value of T, the fundamental angular frequency, $\omega_0 = 2\pi/T$, becomes a very small quantity, say $\Delta\omega$. Also, the angular frequencies corresponding to $\omega_n = n\cdot\omega_0 = n\cdot\Delta\omega$, $(n = 1, 2, \ldots, \infty)$, now take values closer and closer to each other, suggesting the need for a continuous spectrum of values.

The non-periodic function can be written, therefore, as

$$f(x) = \int_0^\infty [C(\omega)\cdot\cos(\omega\cdot x) + S(\omega)\cdot\sin(\omega\cdot x)]d\omega,$$

where

$$C(\omega) = \frac{1}{2\pi}\cdot\int_{-\infty}^{\infty} f(x)\cdot\cos(\omega\cdot x)\cdot dx,$$

and

$$S(\omega) = \frac{1}{2\pi}\cdot\int_{-\infty}^{\infty} f(x)\cdot\sin(\omega\cdot x)\cdot dx$$

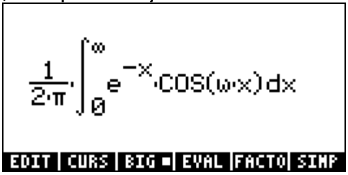
The continuous spectrum is given by

$$A(\omega) = \sqrt{[C(\omega)]^2 + [S(\omega)]^2}$$

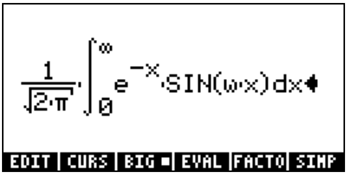
The functions $C(\omega)$, $S(\omega)$, and $A(\omega)$ are continuous functions of a variable $\omega$, which becomes the transform variable for the Fourier transforms defined below.

<u>Example 1</u> – Determine the coefficients $C(\omega)$, $S(\omega)$, and the continuous spectrum $A(\omega)$, for the function $f(x) = \exp(-x)$, for $x > 0$, and $f(x) = 0$, $x < 0$.

In the calculator, set up and evaluate the following integrals to calculate C(ω) and S(ω), respectively. CAS modes are set to Exact and Real.

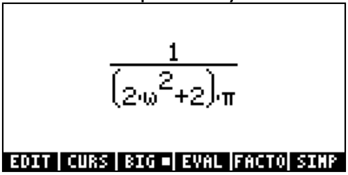


$$\frac{1}{2\cdot\pi}\cdot\int_{0}^{\infty}e^{-x}\cdot\mathrm{COS}(\omega\cdot x)\,dx$$

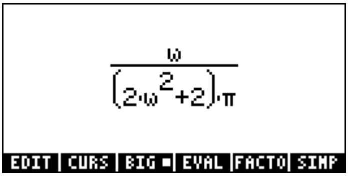


$$\frac{1}{\sqrt{2\cdot\pi}}\cdot\int_{0}^{\infty}e^{-x}\cdot\mathrm{SIN}(\omega\cdot x)\,dx$$

Their results are, respectively:

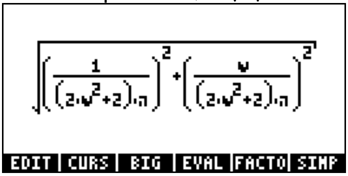


$$\frac{1}{\left(2\cdot\omega^{2}+2\right)\cdot\pi}$$

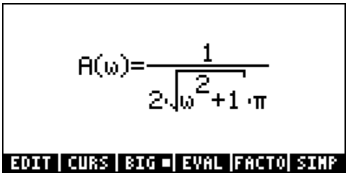


$$\frac{\omega}{\left(2\cdot\omega^{2}+2\right)\cdot\pi}$$

The continuous spectrum, A(ω) is calculated as:

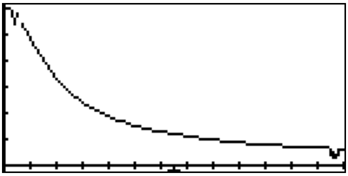


$$\sqrt{\left(\frac{1}{\left(2\cdot\omega^{2}+2\right)\cdot\pi}\right)^{2}+\left(\frac{\omega}{\left(2\cdot\omega^{2}+2\right)\cdot\pi}\right)^{2}}$$



$$A(\omega)=\frac{1}{2\cdot\sqrt{\omega^{2}+1}\cdot\pi}$$

Define this expression as a function by using function DEFINE (↰ DEF ). Then, plot the continuous spectrum, in the range 0 < ω < 10, as:



## Definition of Fourier transforms

Different types of Fourier transforms can be defined. The following are the definitions of the sine, cosine, and full Fourier transforms and their inverses used in this Chapter:

Fourier sine transform

$$\mathsf{F}_s\{f(t)\} = F(\omega) = \frac{2}{\pi}\cdot\int_0^\infty f(t)\cdot\sin(\omega\cdot t)\cdot dt$$

Inverse sine transform

$$\mathsf{F}_s^{-1}\{F(\omega)\} = f(t) = \int_0^\infty F(\omega)\cdot\sin(\omega\cdot t)\cdot dt$$

Fourier cosine transform

$$\mathsf{F}_c\{f(t)\} = F(\omega) = \frac{2}{\pi}\cdot\int_0^\infty f(t)\cdot\cos(\omega\cdot t)\cdot dt$$

Inverse cosine transform

$$\mathsf{F}_c^{-1}\{F(\omega)\} = f(t) = \int_0^\infty F(\omega)\cdot\cos(\omega\cdot t)\cdot dt$$

Fourier transform (proper)

$$\mathsf{F}\{f(t)\} = F(\omega) = \frac{1}{2\pi}\cdot\int_{-\infty}^\infty f(t)\cdot e^{-i\omega t}\cdot dt$$

Inverse Fourier transform (proper)

$$\mathsf{F}^{-1}\{F(\omega)\} = f(t) = \int_{-\infty}^\infty F(\omega)\cdot e^{-i\omega t}\cdot dt$$

<u>Example 1</u> – Determine the Fourier transform of the function f(t) = exp(- t), for t >0, and f(t) = 0, for t<0.

The continuous spectrum, F(ω), is calculated with the integral:

$$\frac{1}{2\pi}\int_0^\infty e^{-(1+i\omega)t}dt = \lim_{\varepsilon\to\infty}\frac{1}{2\pi}\int_0^\varepsilon e^{-(1+i\omega)t}dt$$

$$= \lim_{\varepsilon\to\infty}\frac{1}{2\pi}\left[\frac{1-\exp(-(1+i\omega)\varepsilon)}{1+i\omega}\right] = \frac{1}{2\pi}\cdot\frac{1}{1+i\omega}.$$

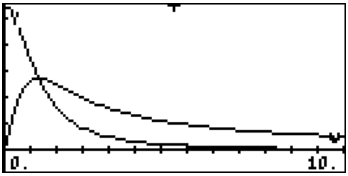
This result can be rationalized by multiplying numerator and denominator by the conjugate of the denominator, namely, 1-iω. The result is now:

$$F(\omega)=\frac{1}{2\pi}\cdot\frac{1}{1+i\omega}=\frac{1}{2\pi}\cdot\left(\frac{1}{1+i\omega}\right)\cdot\left(\frac{1-i\omega}{1-i\omega}\right)$$

$$=\frac{1}{2\pi}\left(\frac{1}{1+\omega^2}-i\cdot\frac{\omega}{1+\omega^2}\right)$$

which is a complex function.

The absolute value of the real and imaginary parts of the function can be plotted as shown below



**Notes**:
The magnitude, or absolute value, of the Fourier transform, |F(ω)|, is the frequency spectrum of the original function f(t). For the example shown above, $|F(\omega)| = 1/[2\pi(1+\omega^2)]^{1/2}$. The plot of |F(ω)| vs. ω was shown earlier.

Some functions, such as constant values, sin x, exp(x), $x^2$, etc., do not have Fourier transform. Functions that go to zero sufficiently fast as x goes to infinity do have Fourier transforms.

## Properties of the Fourier transform

Linearity: If a and b are constants, and f and g functions, then F{a·f + b·g} = a F{f }+ b F{g}.

Transformation of partial derivatives. Let u = u(x,t). If the Fourier transform transforms the variable x, then

$$F\{\partial u/\partial x\} = i\omega\ F\{u\},\ F\{\partial^2 u/\partial x^2\} = -\omega^2\ F\{u\},$$
$$F\{\partial u/\partial t\} = \partial F\{u\}/\partial t,\ F\{\partial^2 u/\partial t^2\} = \partial^2 F\{u\}/\partial t^2$$

convolution: For Fourier transform applications, the operation of convolution is defined as

$$(f * g)(x) = \frac{1}{\sqrt{2\pi}} \cdot \int f(x-\xi) \cdot g(\xi) \cdot d\xi.$$

The following property holds for convolution:

$$F\{f*g\} = F\{f\} \cdot F\{g\}.$$

## Fast Fourier Transform (FFT)

The Fast Fourier Transform is a computer algorithm by which one can calculate very efficiently a discrete Fourier transform (DFT). This algorithm has applications in the analysis of different types of time-dependent signals, from turbulence measurements to communication signals.

The discrete Fourier transform of a sequence of data values $\{x_j\}$, j = 0, 1, 2, …, n-1, is a new finite sequence $\{X_k\}$, defined as

$$X_k = \frac{1}{n} \sum_{j=0}^{n-1} x_j \cdot \exp(-i \cdot 2\pi kj / n), \qquad k = 0,1,2,...,n-1$$

The direct calculation of the sequence $X_k$ involves $n^2$ products, which would involve enormous amounts of computer (or calculator) time particularly for large values of n. The Fast Fourier Transform reduces the number of operations to the order of $n \cdot \log_2 n$. For example, for n = 100, the FFT requires about 664 operations, while the direct calculation would require 10,000 operations. Thus, the number of operations using the FFT is reduced by a factor of 10000/664 ≈ 15.

The FFT operates on the sequence $\{x_j\}$ by partitioning it into a number of shorter sequences. The DFT's of the shorter sequences are calculated and later combined together in a highly efficient manner. For details on the algorithm refer, for example, to Chapter 12 in Newland, D.E., 1993, "An

Introduction to Random Vibrations, Spectral & Wavelet Analysis – Third Edition," Longman Scientific and Technical, New York.

The only requirement for the application of the FFT is that the number n be a power of 2, i.e., select your data so that it contains 2, 4, 8, 16, 32, 62, etc., points.

## Examples of FFT applications

FFT applications usually involve data discretized from a time-dependent signal. The calculator can be fed that data, say from a computer or a data logger, for processing. Or, you can generate your own data by programming a function and adding a few random numbers to it.

Example 1 – Define the function f(x) = 2 sin (3x) + 5 cos(5x) + 0.5*RAND, where RAND is the uniform random number generator provided by the calculator. Generate 128 data points by using values of x in the interval (0,12.8). Store those values in an array, and perform a FFT on the array.

First, we define the function f(x) as a RPN program:

```
<< → x '2*SIN(3*x) + 5*COS(5*x)' EVAL RAND 5 * + →NUM >>
```

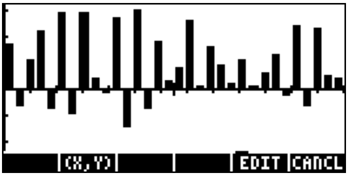
and store this program in variable f. Next, type the following program to generate $2^m$ data values between a and b. The program will take the values of m, a, and b:

```
<< → m a b << '2^m' EVAL → n << '(b-a)/(n+1)' EVAL → Dx << 1 n FOR j
'a+(j-1)*Dx' EVAL f  NEXT  n →ARRY >>  >>  >>  >>
```

Store this program under the name GDATA (Generate DATA). Then, run the program for the values, m = 5, a = 0, b = 100. In RPN mode, use:

5 SPC 0 SPC 1 0 0 GDATA

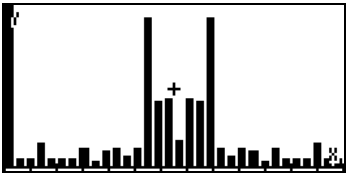
The figure below is a box plot of the data produced. To obtain the graph, first copy the array just created, then transform it into a column vector by using: OBJ→ 1 + →ARRY (Functions OBJ→ and →ARRY are available in the command catalog, → _CAT ). Store the array into variable ΣDAT by using function STOΣ (also available through → _CAT ). Select Bar in the TYPE for graphs, change the view window to H-VIEW: 0 32, V-VIEW: -10 10, and BarWidth to 1. Press CANCL ON to return to normal calculator display.



To perform the FFT on the array in stack level 1 use function FFT available in the MTH/FFT menu on array ΣDAT: ΣDAT FFT. The FFT returns an array of complex numbers that are the arrays of coefficients $X_k$ of the DFT. The magnitude of the coefficients $X_k$ represents a frequency spectrum of the original data. To obtain the magnitude of the coefficients you could transform the array into a list, and then apply function ABS to the list. This is accomplished by using: OBJ→ EVAL ◄ →LIST ← ABS

Finally, you can convert the list back to a column vector to be stored in ΣDAT, as follows: OBJ→ 1 ENTER 2 →LIST →ARRY STOΣ

To plot the spectrum, follow the instructions for producing a bar plot given earlier. The vertical range needs to be changed to –1 to 80. The spectrum of frequencies is the following:



The spectrum shows two large components for two frequencies (these are the sinusoidal components, sin (3x) and cos(5x)), and a number of smaller components for other frequencies.

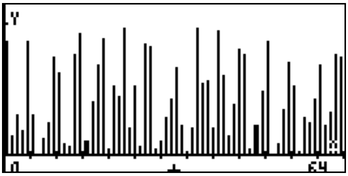
Example 2 – To produce the signal given the spectrum, we modify the program GDATA to include an absolute value, so that it reads:

```
<< → m a b << '2^m' EVAL → n << '(b-a)/(n+1)' EVAL → Dx << 1 n FOR j
'a+(j-1)*Dx' EVAL f  ABS NEXT  n →ARRY >>  >>  >>  >>
```

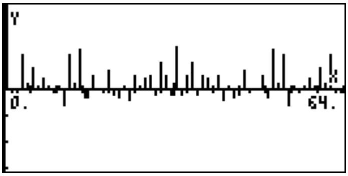
Store this version of the program under GSPEC (Generate SPECtrum). Run the program with m = 6, a = 0, b = 100. In RPN mode, use:

6 SPC 0 SPC 1 0 0 [illegible]

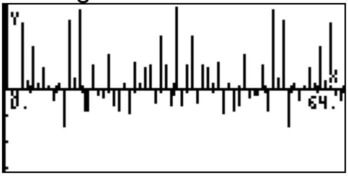
Press ENTER when done, to keep an additional copy of the spectrum array. Convert this row vector into a column vector and store it into ΣDAT. Following the procedure for generating a bar plot, the spectrum generated for this example looks as shown below. The horizontal range in this case is 0 to 64, while the vertical range is –1 to 10:



To reproduce the signal whose spectrum is shown , use function IFFT. Since we left a copy of the spectrum in the stack (a row vector), all you need to do if find function IFFT in the MTH/FFT menu or through the command catalog, → CAT . As an alternative, you could simply type the function name, i.e., type ALPHA ALPHA I F F T ENTER . The signal is shown as an array (row vector) with complex numbers. We are interested only in the real part of the elements. To extract the real part of the complex numbers, use function RE from the CMPLX menu (see Chapter 4), e.g., type ALPHA ALPHA R E ENTER . What results is another row vector. Convert it into a column vector, store it into SDAT, and plot a bar plot to show the signal. The signal for this example is shown below, using a horizontal range of 0 to 64, and a vertical range of –1 to 1:



Except for a large peak at t = 0, the signal is mostly noise. A smaller vertical scale (-0.5 to 0.5) shows the signal as follows:



# Solution to specific second-order differential equations

In this section we present and solve specific types of ordinary differential equations whose solutions are defined in terms of some classical functions, e.g., Bessel's functions, Hermite polynomials, etc. Examples are presented in RPN mode.

## The Cauchy or Euler equation

An equation of the form $x^2 \cdot (d^2y/dx^2) + a \cdot x \cdot (dy/dx) + b \cdot y = 0$, where a and b are real constants, is known as the Cauchy or Euler equation. A solution to the Cauchy equation can be found by assuming that $y(x) = x^n$.
Type the equation as: 'x^2*d1d1y(x)+a*x*d1y(x)+b*y(x)=0' ENTER
Then, type and substitute the suggested solution: 'y(x) = x^n' ENTER SUBST

The result is: 'x^2*(n*(x^(n-1-1)*(n-1)))+a*x*(n*x^(n-1))+b*x^n =0, which simplifies to 'n*(n-1)*x^n+a*n*x^n+b*x^n = 0'. Dividing by x^n, results in an auxiliary algebraic equation: 'n*(n-1)+a*n+b = 0', or.

$$n^2 + (a-1) \cdot n + b = 0 .$$

- If the equation has two different roots, say $n_1$ and $n_2$, then the general solution of this equation is $y(x) = K_1 \cdot x\ ^n{}_1 + K_2 \cdot x\ ^n{}_2$.
- If $b = (1-a)^2/4$, then the equation has a double root $n_1 = n_2 = n = (1-a)/2$, and the solution turns out to be $y(x) = (K_1 + K_2 \cdot \ln x)x^n$.

## Legendre's equation

An equation of the form $(1-x^2)\cdot(d^2y/dx^2)-2\cdot x\cdot (dy/dx)+n\cdot (n+1) \cdot y = 0$, where n is a real number, is known as the Legendre's differential equation. Any solution for this equation is known as a Legendre's function. When n is a nonnegative integer, the solutions are called Legendre's polynomials. Legendre's polynomial of order n is given by

$$P_n(x) = \sum_{m=0}^{M} (-1)^m \cdot \frac{(2n-2m)!}{2^n \cdot m! \cdot (n-m)! \cdot (n-2m)!} \cdot x^{n-2m}$$

$$= \frac{(2n)!}{2^n \cdot (n!)^2} \cdot x^n - \frac{(2n-2)!}{2^n \cdot 1! \cdot (n-1)!(n-2)!} \cdot x^{n-2} + \ldots - ..$$

where M = n/2 or (n-1)/2, whichever is an integer.

Legendre's polynomials are pre-programmed in the calculator and can be recalled by using the function LEGENDRE given the order of the polynomial, n. The function LEGENDRE can be obtained from the command catalog (↱ CAT ) or through the menu ARITHMETIC/POLYNOMIAL menu (see Chapter 5). In RPN mode, the first six Legendre polynomials are obtained as follows:

0 LEGENDRE, result: 1, i.e., $P_0(x) = 1.0$.
1 LEGENDRE, result: 'X', i.e., $P_1(x) = x$.
2 LEGENDRE, result: '(3*X^2-1)/2', i.e., $P_2(x) = (3x^2-1)/2$.
3 LEGENDRE, result: '(5*X^3-3*X)/2', i.e., $P_3(x) =(5x^3-3x)/2$.
4 LEGENDRE, result: '(35*X^4-30*X^2+3)/8', i.e.,
$P_4(x) =(35x^4-30x^2+3)/8$.
5 LEGENDRE, result: '(63*X^5-70*X^3+15*X)/8', i.e.,
$P_5(x) =(63x^5-70x^3+15x)/8$.

The ODE $(1-x^2)\cdot(d^2y/dx^2)-2\cdot x\cdot (dy/dx)+[n\cdot (n+1)-m^2/(1-x^2)] \cdot y = 0$, has for solution the function $y(x) = P_n{}^m(x)= (1-x^2)^{m/2}\cdot(d^mPn/dx^m)$. This function is referred to as an associated Legendre function.

## Bessel's equation

The ordinary differential equation $x^2\cdot(d^2y/dx^2) + x\cdot (dy/dx)+ (x^2-\nu^2)\cdot y = 0$, where the parameter ν is a nonnegative real number, is known as Bessel's differential equation. Solutions to Bessel's equation are given in terms of Bessel functions of the first kind of order ν:

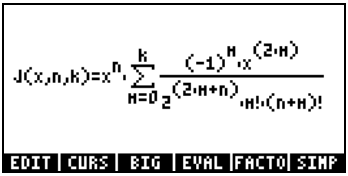
$$J_\nu(x) = x^\nu \cdot \sum_{m=0}^{\infty} \frac{(-1)^m \cdot x^{2m}}{2^{2m+\nu} \cdot m!\cdot\Gamma(\nu+m+1)},$$

where ν is not an integer, and the function Gamma Γ(α) is defined in Chapter 3.

If ν = n, an integer, the Bessel functions of the first kind for n = integer are defined by

$$J_n(x) = x^n \cdot \sum_{m=0}^{\infty} \frac{(-1)^m \cdot x^{2m}}{2^{2m+n} \cdot m!\cdot(n+m)!}.$$

Regardless of whether we use ν (non-integer) or n (integer) in the calculator, we can define the Bessel functions of the first kind by using the following finite series:



$$J(x,n,k)=x^n\cdot\sum_{m=0}^{k}\frac{(-1)^m\cdot x^{(2\cdot m)}}{2^{(2\cdot m+n)}\cdot m!\cdot(n+m)!}$$

Thus, we have control over the function's order, n, and of the number of elements in the series, k. Once you have typed this function, you can use function DEFINE to define function J(x,n,k). This will create the variable J in the soft-menu keys. For example, to evaluate $J_3(0.1)$ using 5 terms in the series, calculate J(0.1,3,5), i.e., in RPN mode: • 1 SPC 3 SPC 5 J The result is 2.08203157E-5.

If you want to obtain an expression for $J_0(x)$ with, say, 5 terms in the series, use J(x,0,5). The result is

‘1-0.25*x^3+0.015625*x^4-4.3403777E-4*x^6+6.782168E-6*x^8-6.78168*x^10’.

For non-integer values ν, the solution to the Bessel equation is given by

$$y(x) = K_1 \cdot J_\nu(x)+K_2 \cdot J_{-\nu}(x).$$

For integer values, the functions Jn(x) and J-n(x) are linearly dependent, since

$$J_n(x) = (-1)^n \cdot J_{-n}(x),$$

therefore, we cannot use them to obtain a general function to the equation. Instead, we introduce the Bessel functions of the second kind defined as

$$Y_\nu(x) = [J_\nu(x) \cos \nu\pi - J_{-\nu}(x)]/\sin \nu\pi,$$

for non-integer ν, and for n integer, with n > 0, by

$$Y_n(x) = \frac{2}{\pi} \cdot J_n(x) \cdot (\ln\frac{x}{2} + \gamma) + \frac{x^n}{\pi} \cdot \sum_{m=0}^{\infty} \frac{(-1)^{m-1} \cdot (h_m + h_{m+n})}{2^{2m+n} \cdot m! \cdot (m+n)!} \cdot x^{2m}$$

$$-\frac{x^{-n}}{\pi} \cdot \sum_{m=0}^{n-1} \frac{(n-m-1)!}{2^{2m-n} \cdot m!} \cdot x^{2m}$$

where γ is the Euler constant, defined by

$$\gamma = \lim_{r\to\infty}[1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{r} - \ln r] \approx 0.57721566490\ldots,$$

and $h_m$ represents the harmonic series

$$h_m = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{m}$$

For the case n = 0, the Bessel function of the second kind is defined as

$$Y_0(x) = \frac{2}{\pi} \cdot \left[ J_0(x) \cdot (\ln\frac{x}{2} + \gamma) + \sum_{m=0}^{\infty} \frac{(-1)^{m-1} \cdot h_m}{2^{2m} \cdot (m!)^2} \cdot x^{2m} \right].$$

With these definitions, a general solution of Bessel's equation for all values of $\nu$ is given by $y(x) = K_1 \cdot J_\nu(x)+K_2 \cdot Y_\nu(x)$.

In some instances, it is necessary to provide complex solutions to Bessel's equations by defining the Bessel functions of the third kind of order $\nu$ as

$$H_n^{(1)}(x) = J_\nu(x)+i \cdot Y_\nu(x), \text{ and } H_n^{(2)}(x) = J_\nu(x)-i \cdot Y_\nu(x),$$

These functions are also known as the first and second Hankel functions of order $\nu$.

In some applications you may also have to utilize the so-called modified Bessel functions of the first kind of order $\nu$ defined as $I_\nu(x)= i^{-\nu} \cdot J_\nu(i \cdot x)$, where i is the unit imaginary number. These functions are solutions to the differential equation $x^2 \cdot (d^2y/dx^2) + x \cdot (dy/dx)\text{-} (x^2+\nu^2) \cdot y = 0$.

The modified Bessel functions of the second kind,

$$K_\nu(x) = (\pi/2) \cdot [I_{-\nu} (x)-I_\nu (x)]/\sin \nu\pi,$$

are also solutions of this ODE.

You can implement functions representing Bessel's functions in the calculator in a similar manner to that used to define Bessel's functions of the first kind, but keeping in mind that the infinite series in the calculator need to be translated into a finite series.

## Chebyshev or Tchebycheff polynomials

The functions $T_n(x) = \cos(n \cdot \cos^{-1} x)$, and $U_n(x) = \sin[(n+1) \cos^{-1} x]/(1\text{-}x^2)^{1/2}$, n = 0, 1, … are called Chebyshev or Tchebycheff polynomials of the first and second kind, respectively. The polynomials Tn(x) are solutions of the differential equation $(1\text{-}x^2) \cdot (d^2y/dx^2) - x \cdot (dy/dx) + n^2 \cdot y = 0$.

In the calculator the function TCHEBYCHEFF generates the Chebyshev or Tchebycheff polynomial of the first kind of order n, given a value of n > 0. If the integer n is negative (n < 0), the function TCHEBYCHEFF generates a Tchebycheff polynomial of the second kind of order n whose definition is

$$U_n(x) = \sin(n\cdot\arccos(x))/\sin(\arccos(x)).$$

You can access the function TCHEBYCHEFF through the command catalog (↱ _CAT_ ).

The first four Chebyshev or Tchebycheff polynomials of the first and second kind are obtained as follows:

| | | |
|---|---|---|
| 0 TCHEBYCHEFF, result: 1, | i.e., | $T_0(x) = 1.0.$ |
| -0 TCHEBYCHEFF, result: 1, | i.e., | $U_0(x) = 1.0.$ |
| 1 TCHEBYCHEFF, result: ‘X’, | i.e., | $T_1(x) = x.$ |
| -1 TCHEBYCHEFF, result: 1, | i.e., | $U_1(x) =1.0.$ |
| 2 TCHEBYCHEFF, result: ‘2*X^2-1, | i.e., | $T_2(x) =2x^2-1.$ |
| -2 TCHEBYCHEFF, result: ‘2*X’, | i.e., | $U_2(x) =2x.$ |
| 3 TCHEBYCHEFF, result: ‘4*X^3-3*X’, | i.e., | $T_3(x) = 4x^3-3x.$ |
| -3 TCHEBYCHEFF, result: ‘4*X^2-1’, | i.e., | $U_3(x) = 4x^2-1.$ |

## Laguerre’s equation

Laguerre’s equation is the second-order, linear ODE of the form $x\cdot(d^2y/dx^2) +(1-x)\cdot (dy/dx) + n\cdot y = 0$. Laguerre polynomials, defined as

$$L_0(x) = 1, \quad L_n(x) = \frac{e^x}{n!}\cdot\frac{d^n(x^n\cdot e^{-x})}{dx^n}, n = 1,2,\ldots,$$

are solutions to Laguerre’s equation. Laguerre’s polynomials can also be calculated with:

$$L_n(x) = \sum_{m=0}^{n}\frac{(-1)^m}{m!}\cdot\begin{pmatrix} n \\ m \end{pmatrix}\cdot x^m.$$

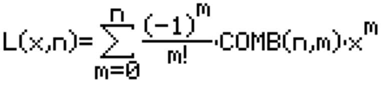
$$= 1 - n\cdot x + \frac{n(n-1)}{4}\cdot x^2 - \ldots + \ldots. + \frac{(-1)^n}{n!}\cdot x^n$$

The term

$$\begin{pmatrix} n \\ m \end{pmatrix} = \frac{n!}{m!(n-m)!} = C(n,m)$$

is the m-th coefficient of the binomial expansion $(x+y)^n$. It also represents the number of combinations of n elements taken m at a time. This function is available in the calculator as function COMB in the MTH/PROB menu (see also Chapter 17).

You can define the following function to calculate Laguerre's polynomials:



$$L(x,n)=\sum_{m=0}^{n}\frac{(-1)^m}{m!}\cdot COMB(n,m)\cdot x^m$$

When done typing it in the equation writer press use function DEFINE to create the function L(x,n) into variable L .

To generate the first four Laguerre polynomials use, L(x,0), L(x,1), L(x,2), L(x,3). The results are:

$L_0(x)$ = .
$L_1(x)$ = 1-x.
$L_2(x)$ = 1-2x+ $0.5x^2$
$L_3(x)$ = 1-3x+$1.5x^2$-0.16666…$x^3$.

## Weber's equation and Hermite polynomials

Weber's equation is defined as $d^2y/dx^2+(n+1/2-x^2/4)y = 0$, for n = 0, 1, 2, … A particular solution of this equation is given by the function , $y(x) = \exp(-x^2/4)H^*(x/\sqrt{2})$, where the function $H^*(x)$ is the Hermite polynomial:

$$H_0* = 1, \quad H_n*(x) = (-1)^n e^{x^2}\frac{d^n}{dx^n}(e^{-x^2}), \quad n = 1,2,..$$

In the calculator, the function HERMITE, available through the menu ARITHMETIC/POLYNOMIAL. Function HERMITE takes as argument an integer number, n, and returns the Hermite polynomial of n-th degree. For example, the first four Hermite polynomials are obtained by using:

0 HERMITE, result: 1,  i.e., $H_0^* = 1$.
1 HERMITE, result: '2*X',  i.e., $H_1^* = 2x$.
2 HERMITE, result: '4*X^2-2',  i.e., $H_2^* = 4x^2-2$.
3 HERMITE, result: '8*X^3-12*X',  i.e., $H_3^* = 8x^3-12x$.

## Numerical and graphical solutions to ODEs

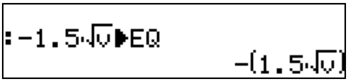
Differential equations that cannot be solved analytically can be solved numerically or graphically as illustrated below.

### Numerical solution of first-order ODE

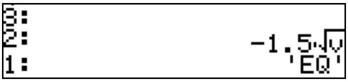
Through the use of the numerical solver (↱ NUM.SLV), you can access an input form that lets you solve first-order, linear ordinary differential equations. The use of this feature is presented using the following example. The method used in the solution is a fourth-order Runge-Kutta algorithm preprogrammed in the calculator.

Example 1 -- Suppose we want to solve the differential equation, dv/dt = -1.5 $v^{1/2}$, with v = 4 at t = 0. We are asked to find v for t = 2.

First, create the expression defining the derivative and store it into variable EQ. The figure to the left shows the ALG mode command, while the right-hand side figure shows the RPN stack before pressing STO▶.

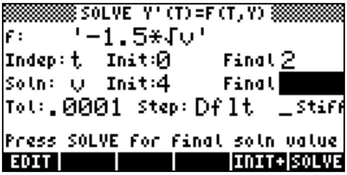


```
:-1.5·√v▶EQ
                 -(1.5·√v)
```



```
3:
2:               -1.5·√v
1:                  'EQ'
```

Then, enter the NUMERICAL SOLVER environment and select the differential equation solver: ↱ NUM.SLV ▼ OK . Enter the following parameters:



```
SOLVE Y'(T)=F(T,Y)
F:    '-1.5*√v'
Indep: t   Init:0     Final 2
Soln:  v   Init:4     Final
Tol:.0001  Step: Dflt  _Stiff
Press SOLVE for final soln value
```

To solve, press: SOLVE (wait) EDIT. The result is 0.2499 ≈ 0.25. Press OK.

**Solution presented as a table of values**

Suppose we wanted to produce a table of values of v, for t = 0.00, 0.25, …, 2.00, we will proceed as follows:

First, prepare a table to write down your results. Write down in your table the step-by-step results:

| t | v |
|---|---|
| 0.00 | 0.00 |
| 0.25 | |
| … | … |
| 2.00 | |

Next, within the SOLVE environment, change the final value of the independent variable to 0.25, use :

▲ .25 OK ▶ ▶ SOLVE (wait) EDIT
(Solves for v at t = 0.25, v = 3.285 …. )
OK INIT+ ▲ . 5 OK ▶ ▶ SOLVE (wait) EDIT
(Changes initial value of t to 0.25, and final value of t to 0.5, solve for v(0.5) = 2.640…)
OK INIT+ ▲ .75 OK ▶ ▶ SOLVE (wait) EDIT
(Changes initial value of t to 0.5, and final value of t to 0.75, solve for v(0.75) = 2.066…)
OK INIT+ ▲ 1 OK ▶ ▶ SOLVE (wait) EDIT
(Changes initial value of t to 0.75, and final value of t to 1, solve for v(1) = 1.562…)
Repeat for t = 1.25, 1.50, 1.75, 2.00. Press OK after viewing the last result in EDIT. To return to normal calculator display, press ON or NXT OK. The different solutions will be shown in the stack, with the latest result in level 1.

The final results look as follows (rounded to the third decimal):

| **t** | **v** |
|---|---|

| | |
|---|---|
| 0.00 | 4.000 |
| 0.25 | 3.285 |
| 0.50 | 2.640 |
| 0.75 | 2.066 |
| 1.00 | 1.562 |
| 1.25 | 1.129 |
| 1.50 | 0.766 |
| 1.75 | 0.473 |
| 2.00 | 0.250 |

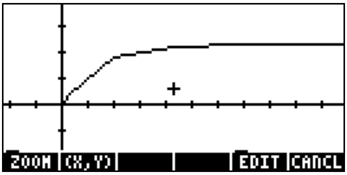
## Graphical solution of first-order ODE

When we can not obtain a closed-form solution for the integral, we can always plot the integral by selecting `Diff Eq` in the `TYPE` field of the PLOT environment as follows: suppose that we want to plot the position x(t) for a velocity function v(t) = exp(-t$^2$), with x = 0 at t = 0. We know there is no closed-form expression for the integral, however, we know that the definition of v(t) is dx/dt = exp(-t$^2$).

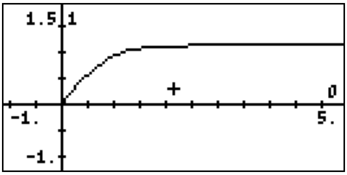
The calculator allows for the plotting of the solution of differential equations of the form Y'(T) = F(T,Y). For our case, we let Y = x and T = t, therefore, F(T,Y) = f(t, x) = exp(-t$^2$). Let's plot the solution, x(t), for t = 0 to 5, by using the following keystroke sequence:

- ↰ 2D/3D (simultaneously, if in RPN mode) to enter PLOT environment
- Highlight the field in front of TYPE, using the ▲▼ keys. Then, press CHOOS, and highlight `Diff Eq`, using the ▲▼ keys. Press OK.
- Change field F: to 'EXP(- t^2)'
- Make sure that the following parameters are set to: `H-VAR: 0, V-VAR: 1`
- Change the independent variable to t .
- Accept changes to PLOT SETUP: NXT OK
- ↰ WIN (simultaneously, if in RPN mode). To enter PLOT WINDOW environment
- Change the horizontal and vertical view window to the following settings: `H-VIEW: -1    5;  V-VIEW: -1     1.5`

- Also, use the following values for the remaining parameters: Init: 0, Final: 5, Step: Default, Tol: 0.0001, Init-Soln: 0
- To plot the graph use: ERASE DRAW



When you observe the graph being plotted, you'll notice that the graph is not very smooth. That is because the plotter is using a time step that may be a bit large for a smooth graph. To refine the graph and make it smoother, use a step of 0.1. Press CANCL and change the *Step* : value to 0.1, then use ERASE DRAW once more to repeat the graph. The plot will take longer to be completed, but the shape is definitely smoother than before. Try the following: EDIT NXT LABEL MENU to see axes labels and range.



Notice that the labels for the axes are shown as 0 (horizontal, for t) and 1 (vertical, for x). These are the definitions for the axes as given in the PLOT SETUP window (← 2D/3D ) i.e., H-VAR: 0, and V-VAR: 1. To see the graphical solution in detail use the following:

| | |
|---|---|
| NXT NXT PICT | To recover menu and return to PICT environment. |
| (X,Y) | To determine coordinates of any point on the graph. |

Use the ◀ ▶ keys to move the cursor around the plot area. At the bottom of the screen you will see the coordinates of the cursor as (X,Y), i.e., the calculator uses X and Y as the default names for the horizontal and vertical axes, respectively. Press NXT CANCL to recover the menu and return to the PLOT WINDOW environment. Finally, press ON to return to normal display.

## Numerical solution of second-order ODE

Integration of second-order ODEs can be accomplished by defining the solution as a vector. As an example, suppose that a spring-mass system is subject to a damping force proportional to its speed, so that the resulting differential equation is: $\frac{d^2x}{dt^2} = -18.75 \cdot x - 1.962 \cdot \frac{dx}{dt}$

or, $x'' = -18.75\ x - 1.962\ x'$,

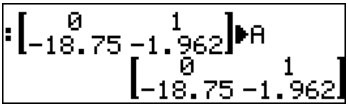
subject to the initial conditions, v = x' = 6, x = 0, at t = 0. We want to find x, x' at t = 2.

Re-write the ODE as: **w**' = **Aw**, where **w** = [ x x' ]$^T$, and **A** is the 2 x 2 matrix shown below.

$$\begin{bmatrix} x \\ x' \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -18.75 & -1.962 \end{bmatrix} \cdot \begin{bmatrix} x \\ x' \end{bmatrix}$$

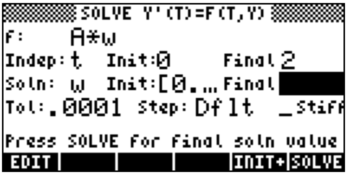
The initial conditions are now written as **w** = [0 6]$^T$, for t = 0. (Note: The symbol [ ]$^T$ means the transpose of the vector or matrix).

To solve this problem, first, create and store the matrix **A**, e.g., in ALG mode:



```
: ⎡   0      1  ⎤
  ⎣-18.75 -1.962⎦▶A
                 ⎡   0      1  ⎤
                 ⎣-18.75 -1.962⎦
```

Then, activate the numerical differential equation solver by using: ↱ NUM.SLV ▼ OK . To solve the differential equation with starting time t = 0 and final time t = 2, the input form for the differential equation solver should look as follows (notice that the Init: value for the Soln: is a vector [0, 6]):



```
SOLVE Y'(T)=F(T,Y)
F:     A*w
Indep: t  Init:0      Final 2
Soln:  w  Init:[0.… Final
Tol:.0001  Step: Dflt   _Stiff

Press SOLVE for final soln value
EDIT |   |   |   | INIT+ | SOLVE
```

Press SOLVE (wait) EDIT to solve for w(t=2). The solution reads [.16716… -.6271…], i.e., x(2) = 0.16716, and x'(2) = v(2) = -0.6271. Press ENTER to return to SOLVE environment.

**Solution presented as a table of values**

In the previous example we were interested only in finding the values of the position and velocity at a given time t. If we wanted to produce a table of values of x and x', for t = 0.00, 0.25, …, 2.00, we will proceed as follows: First, prepare a table to write down your results:

| t | x | x' |
|---|---|---|
| 0.00 | 0.00 | 6.00 |
| 0.25 | | |
| … | … | … |
| 2.00 | | |

Next, within the SOLVE environment, change the final value of the independent variable to 0.25, use:

▲ .25 OK ▶ ▶ SOLVE (wait) EDIT

(Solves for w at t = 0.25, w = [0.968 1.368]. )

OK INIT+ ▲ . 5 OK ▶ ▶ SOLVE (wait) EDIT

(Changes initial value of t to 0.25, and final value of t to 0.5, solve again for w(0.5) = [0.748 -2.616])

OK INIT+ ▲ .75 OK ▶ ▶ SOLVE (wait) EDIT

(Changes initial value of t to 0.5, and final value of t to 0.75, solve again for w(0.75) = [0.0147 -2.859])

OK INIT+ ▲ 1 OK ▶ ▶ SOLVE (wait) EDIT

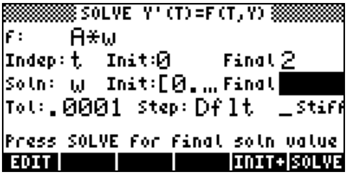
(Changes initial value of t to 0.75, and final value of t to 1, solve again for w(1) = [-0.469  -0.607])

Repeat for t = 1.25, 1.50, 1.75, 2.00. Press OK after viewing the last result in EDIT. To return to normal calculator display, press ON or NXT OK. The different solutions will be shown in the stack, with the latest result in level 1. The final results look as follows:

| t | x | x' | | t | x | x' |
|---|---|---|---|---|---|---|
| 0.00 | 0.000 | 6.000 | | 1.25 | -0.354 | 1.281 |
| 0.25 | 0.968 | 1.368 | | 1.50 | 0.141 | 1.362 |
| 0.50 | 0.748 | -2.616 | | 1.75 | 0.227 | 0.268 |
| 0.75 | -0.015 | -2.859 | | 2.00 | 0.167 | -0.627 |
| 1.00 | -0.469 | -0.607 | | | | |

## Graphical solution for a second-order ODE

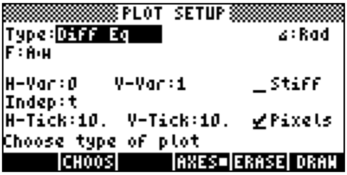
Start by activating the differential equation numerical solver, ↱ NUM.SLV ▼ OK . The SOLVE screen should look like this:



```
SOLVE Y'(T)=F(T,Y)
F:    A*w
Indep: t   Init:0     Final 2
Soln:  w   Init:[0.… Final
Tol:.0001  Step: Dflt  _Stiff
Press SOLVE for final soln value
EDIT |    |    |    |INIT+|SOLVE
```

Notice that the initial condition for the solution (Soln: w Init:[0., …) includes the vector [0, 6]. Press NXT OK.

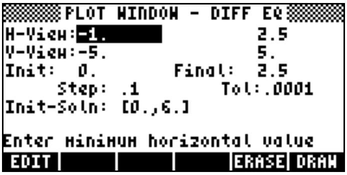
Next, press ↰ 2D/3D (simultaneously, if in RPN mode) to enter the PLOT environment. Highlight the field in front of TYPE, using the ▲▼ keys. Then, press CHOOS, and highlight `Diff Eq`, using the ▲▼ keys. Press OK. Modify the rest of the PLOT SETUP screen to look like this:



```
PLOT SETUP
Type:Diff Eq            ∡:Rad
F:A·W
H-Var:0    V-Var:1      _Stiff
Indep:t
H-Tick:10.  V-Tick:10.  ✓Pixels
Choose type of plot
     |CHOOS|     |AXES■|ERASE|DRAW
```

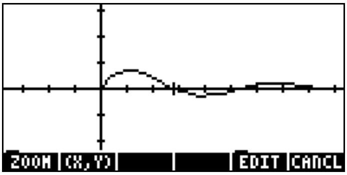
Notice that the option V-Var: is set to 1, indicating that the first element in the vector solution, namely, x', is to be plotted against the independent variable t. Accept changes to PLOT SETUP by pressing NXT OK.

Press ← WIN (simultaneously, if in RPN mode) to enter the PLOT WINDOW environment. Modify this input form to look like this:

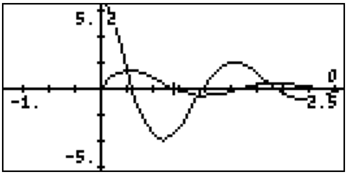


```
PLOT WINDOW - DIFF EQ
H-View:-1.              2.5
V-View:-5.              5.
Init:  0.      Final:   2.5
     Step: .1        Tol:.0001
Init-Soln: [0.,6.]

Enter minimum horizontal value
EDIT |    |    |    |ERASE|DRAW
```

To plot the x' vs. t graph use: ERASE DRAW . The plot of x' vs. t looks like this:



To plot the second curve we need to use the PLOT SETUP input form once, more. To reach this form from the graph above use: CANCL NXT OK ← 2D/3D (simultaneously, if in RPN mode) . Change the value of the V-Var: field to 2, and press DRAW (do not press ERASE or you would loose the graph produced above). Use: EDIT NXT LABEL MENU to see axes labels and range. Notice that the x axis label is the number 0 (indicating the independent variable), while the y-axis label is the number 2 (indicating the second variable, i.e., the last variable plotted). The combined graph looks like this:



Press NXT NXT EXIT CANCL ON to return to normal calculator display.

## Numerical solution for stiff first-order ODE

Consider the ODE: dy/dt = -100y+100t+101, subject to the initial condition y(0) = 1.

### Exact solution

This equation can be written as dy/dt + 100 y = 100 t + 101, and solved using an integrating factor, IF(t) = exp(100t), as follows (RPN mode, with CAS set to Exact mode):

‘(100*t+101)*EXP(100*t)’ ENTER ‘t’ ENTER RISCH

The result is ‘(t+1)*EXP(100*t)’.

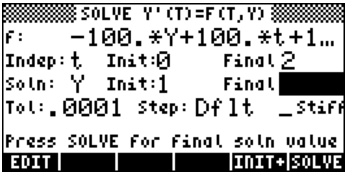
Next, we add an integration constant, by using: ‘C’ ENTER +

Then, we divide by FI(x), by using: ‘EXP(100*t)’ ENTER ÷ .

The result is: ‘((t+1)*EXP(100*t)+C)/EXP(100*t)’, i.e., y(t) = 1+ t +C·e$^{100t}$. Use of the initial condition y(0) = 1, results in 1 = 1 + 0 + C·e$^{0}$, or C = 0, the particular solution being y(t) = 1+t.

### Numerical solution

If we attempt a direct numerical solution of the original equation dy/dt = -100y+100t+101, using the calculator’s own numerical solver, we find that the calculator takes longer to produce a solution that in the previous first-order example. To check this out, set your differential equation numerical solver (↱ NUM.SLV ▼ OK) to:



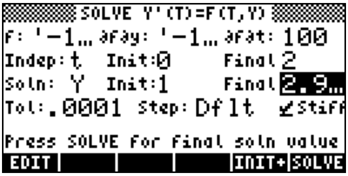
```
SOLVE Y'(T)=F(T,Y)
F:    -100.*Y+100.*t+1…
Indep: t  Init:0     Final 2
Soln:  Y  Init:1     Final
Tol:.0001  Step: Dflt  _Stiff
Press SOLVE for final soln value
EDIT |  |  |  | INIT+ | SOLVE
```

Here we are trying to obtain the value of y(2) given y(0) = 1. With the `Soln: Final` field highlighted, press SOLVE. You can check that a solution takes about 6 seconds, while in the previous first-order example the solution was almost instantaneous. Press ON to cancel the calculation.

This is an example of a stiff ordinary differential equation. A stiff ODE is one whose general solution contains components that vary at widely different rates under the same increment in the independent variable. In this particular case, the general solution, $y(t) = 1+ t +C\cdot e^{100t}$, contains the components 't' and '$C\cdot e^{100t}$', which vary at very different rates, except for the cases C=0 or C≈0 (e.g., for C = 1, t =0.1, $C\cdot e^{100t}$ =22026).

The calculator's ODE numerical solver allows for the solution of stiff ODEs by selecting the option `_Stiff` in the `SOLVE Y'(T) = F(T,Y)` screen. With this option selected you need to provide the values of ∂f/∂y and ∂f/∂t. For the case under consideration ∂f/∂y = -100 and ∂f/∂t = 100.

Enter those values in the corresponding fields of the `SOLVE Y'(T) = F(T,Y)` screen:



```
SOLVE Y'(T)=F(T,Y)
F: '-1… ∂F∂y: '-1… ∂F∂t: 100
Indep: t   Init:0     Final 2
Soln: Y    Init:1     Final 2.9…
Tol:.0001  Step: Dflt  ✓Stiff
Press SOLVE for final soln value
```

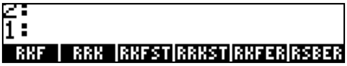
When done, move the cursor to the `Soln:Final` field and press SOLVE. This time, the solution in produced in about 1 second. Press EDIT to see the solution: 2.9999999999, i.e., 3.0.

Note: The option `Stiff` is also available for graphical solutions of differential equations.

## Numerical solution to ODEs with the SOLVE/DIFF menu

The SOLVE soft menu is activated by using 74 MENU in RPN mode. This menu is presented in detail in Chapter 6. One of the sub-menus, DIFF, contains functions for the numerical solution of ordinary differential equations for use in programming. These functions are described next using RPN mode and system flag 117 set to SOFT menus.

The functions provided by the SOLVE/DIFF menu are the following:



## Function RKF

This function is used to compute the solution to an initial value problem for a first-order differential equation using the Runge-Kutta-Fehlbert 4th -5th order solution scheme. Suppose that the differential equation to be solved is given by dy/dx = f(x,y), with y = 0 at x = 0, and that you will allow a convergence criteria ε for the solution. You can also specify an increment in the independent variable, Δx, to be used by the function. To run this function you will prepare your stack as follows:

3: {'x', 'y', 'f(x,y)'}
2: { ε Δx }
1: $x_{final}$

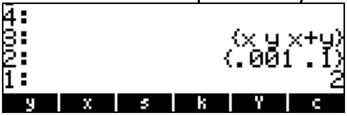
The value in the first stack level is the value of the independent variable where you want to find your solution, i.e., you want to find, $y_{final} = f_s(x_{final})$, where $f_s(x)$ represents the solution to the differential equation. The second stack level may contain only the value of ε, and the step Δx will be taken as a small default value. After running function RKF, the stack will show the lines:

2: {'x', 'y', 'f(x,y)'}
1: ε

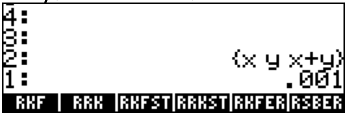
The value of the solution, $y_{final}$, will be available in variable y. This function is appropriate for programming since it leaves the differential equation specifications and the tolerance in the stack ready for a new solution. Notice that the solution uses the initial conditions x = 0 at y = 0. If, your actual initial solutions are $x = x_{init}$ at $y = y_{init}$, you can always add these values to the solution provided by RKF, keeping in mind the following relationship:

| RKF solution | | Actual solution | |
|---|---|---|---|
| x | y | x | y |
| 0 | 0 | $x_{init}$ | $y_{init}$ |
| $x_{final}$ | $y_{final}$ | $x_{init} + x_{final}$ | $y_{init} + y_{final}$ |

The following screens show the RPN stack before and after applying function RKF for the differential equation dy/dx = x+y, ε = 0.001, Δx = 0.1.



```
4:
3:              {x y x+y}
2:              {.001 .1}
1:                      2
  y  |  x  |  ε  |  k  |  Y  |  c
```



```
4:
3:
2:              {x y x+y}
1:                   .001
 RKF | RRK |RKFST|RRKST|RKFER|RSBER
```

After applying function RKF, variable y contains the value 4.3880...

## Function RRK

This function is similar to the RKF function, except that RRK (Rosenbrock and Runge-Kutta methods) requires as the list in stack level 3 for input not only the names of the independent and dependent variables and the function defining the differential equation, but also the expressions for the first and second derivatives of the expression. Thus, the input stack for this function will look as follows:

3: {'x', 'y', 'f(x,y)' '∂f/∂x' '∂f/∂y' }
2: { ε Δx }
1: $x_{final}$

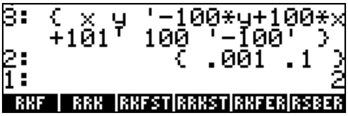
The value in the first stack level is the value of the independent variable where you want to find your solution, i.e., you want to find, $y_{final} = f_s(x_{final})$, where $f_s(x)$ represents the solution to the differential equation. The second stack level may contain only the value of ε, and the step Δx will be taken as a small default value. After running function RRK, the stack will show the lines:

2: {'x', 'y', 'f(x,y)' '∂f/∂x' '∂f/vy' }
1: { ε Δx }

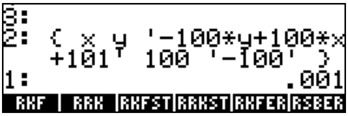
The value of the solution, $y_{final}$, will be available in variable y.

This function can be used to solve so-called "stiff" differential equations.

The following screen shots show the RPN stack before and after application of function RRK:



```
3: { x y '-100*y+100*x
  +101' 100 '-100' }
2:          { .001 .1 }
1:                    2
```



```
3:
2: { x y '-100*y+100*x
  +101' 100 '-100' }
1:                 .001
```

The value stored in variable y is 3.00000000004.

## Function RKFSTEP

This function uses an input list similar to that of function RKF, as well as the tolerance for the solution, and a possible step Δx, and returns the same input list, followed by the tolerance, and an estimate of the next step in the independent variable. The function returns the input list, the tolerance, and the next step in the independent variable that satisfies that tolerance. Thus, the input stack looks as follows:

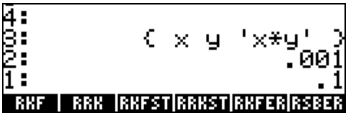
```
3:    {'x', 'y', 'f(x,y)'}
2:                       ε
1:                      Δx
```

After running this function, the stack will show the lines:

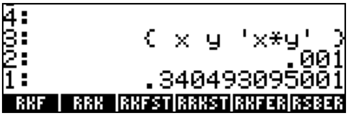
```
3:    {'x', 'y', 'f(x,y)'}
2:                       ε
1:              (Δx)_next
```

Thus, this function is used to determine the appropriate size of a time step to satisfy the required tolerance.

The following screen shots show the RPN stack before and after application of function RKFSTEP:



```
4:
3:        { x y 'x*y' }
2:                 .001
1:                   .1
```



```
4:
3:        { x y 'x*y' }
2:                 .001
1:         .340493095001
```

These results indicate that $(\Delta x)_{next}$ = 0.34049…

## Function RRKSTEP

This function uses an input list similar to that of function RRK, as well as the tolerance for the solution, a possible step Δx, and a number (LAST) specifying the last method used in the solution (1, if RKF was used, or 2, if RRK was used). Function RRKSTEP returns the same input list, followed by the tolerance, an estimate of the next step in the independent variable, and the current method (CURRENT) used to arrive at the next step. Thus, the input stack looks as follows:

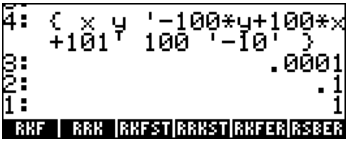
4: {‘x’, ‘y’, ‘f(x,y)’}
3: $\varepsilon$
2: $\Delta x$
1: LAST

After running this function, the stack will show the lines:

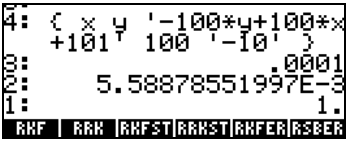
4: {‘x’, ‘y’, ‘f(x,y)’}
3: $\varepsilon$
2: $(\Delta x)_{next}$
1: CURRENT

Thus, this function is used to determine the appropriate size of a time step ($(\Delta x)_{next}$) to satisfy the required tolerance, and the method used to arrive at that result (CURRENT).

The following screen shots show the RPN stack before and after application of function RRKSTEP:



```
4: { x y '-100*y+100*x
   +101' 100 '-10' }
3:                .0001
2:                   .1
1:                    1
```



```
4: { x y '-100*y+100*x
   +101' 100 '-10' }
3:                .0001
2:    5.58878551997E-3
1:                   1.
```

These results indicate that $(\Delta x)_{next}$ = 0.00558… and that the RKF method (CURRENT = 1) should be used.

## Function RKFERR

This function returns the absolute error estimate for a given step when solving a problem as that described for function RKF. The input stack looks as follows:

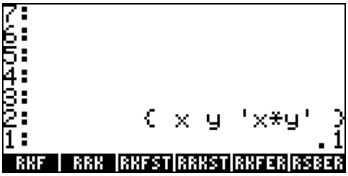
```
2:    {‘x’, ‘y’, ‘f(x,y)’}
1:                      Δx
```

After running this function, the stack will show the lines:

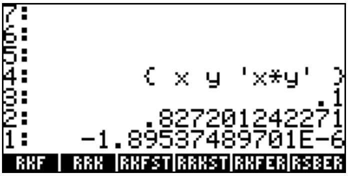
```
4:    {‘x’, ‘y’, ‘f(x,y)’}
3:                       ε
2:                      Δy
1:                   error
```

Thus, this function is used to determine the increment in the solution, Δy, as well as the absolute error (error).

The following screen shots show the RPN stack before and after application of function RKFERR:



```
7:
6:
5:
4:
3:
2:        { x y 'x*y' }
1:                   .1
```



```
7:
6:
5:
4:       { x y 'x*y' }
3:                  .1
2:       .827201242271
1:   -1.89537489701E-6
```

These result show that Δy = 0.827… and error = -1.89…×$10^{-6}$.

## Function RSBERR

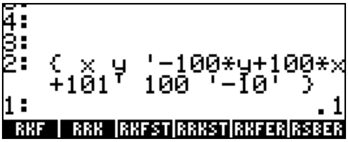
This function performs similarly to RKERR but with the input elements listed for function RRK. Thus, the input stack for this function will look as follows:

```
2:    {‘x’, ‘y’, ‘f(x,y)’  ‘∂f/∂x’ ‘∂f/vy’ }
1:                                      Δx
```

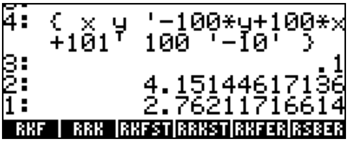
After running the function, the stack will show the lines:

```
4:    {‘x’, ‘y’, ‘f(x,y)’  ‘∂f/∂x’ ‘∂f/vy’ }:
3:                                       ε
2:                                      Δy
1:                                   error
```

The following screen shots show the RPN stack before and after application of function RSBERR:



```
4:
3:
2: { x y '-100*y+100*x
   +101' 100 '-10' }
1:                   .1
```



```
4: { x y '-100*y+100*x
   +101' 100 '-10' }
3:                   .1
2:        4.15144617136
1:        2.76211716614
```

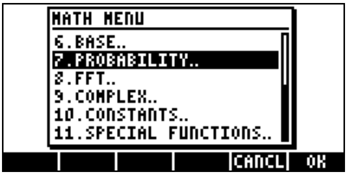
These results indicate that ∆y = 4.1514… and error = 2.762..., for Dx = 0.1. Check that, if Dx is reduced to 0.01, ∆y = -0.00307… and error = 0.000547.

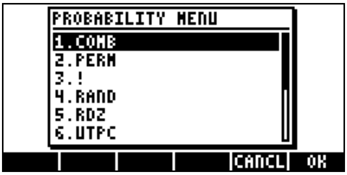
# Chapter 17
# Probability Applications

In this Chapter we provide examples of applications of calculator's functions to probability distributions.

## The MTH/PROBABILITY.. sub-menu - part 1

The MTH/PROBABILITY.. sub-menu is accessible through the keystroke sequence ↰ MTH . With system flag 117 set to CHOOSE boxes, the following list of MTH options is provided (see left-hand side figure below). We have selected the PROBABILITY.. option (option 7), to show the following functions (see right-hand side figure below):





In this section we discuss functions COMB, PERM, ! (factorial), RAND, and RDZ.

### Factorials, combinations, and permutations

The factorial of an integer n is defined as: n! = n· (n-1) · (n-2)…3·2·1. By definition, 0! = 1. Factorials are used in the calculation of the number of permutations and combinations of objects. For example, the number of permutations of r objects from a set of n distinct objects is

$${}_nP_r = n(n-1)(n-1)...(n-r+1) = n!/(n-r)$$

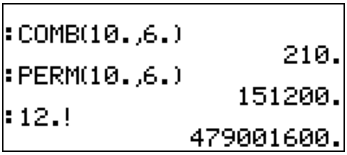
Also, the number of combinations of n objects taken r at a time is

$$\binom{n}{r} = \frac{n(n-1)(n-2)...(n-r+1)}{r!} = \frac{n!}{r!(n-r)!}$$

To simplify notation, use P(n,r) for permutations, and C(n,r) for combinations. We can calculate combinations, permutations, and factorials with functions COMB, PERM, and ! from the MTH/PROBABILITY.. sub-menu. The operation of those functions is described next:

- COMB(n,r): Combinations of n items taken r at a time
- PERM(n,r): Permutations of n items taken r at a time
- n!: Factorial of a positive integer. For a non-integer, x! returns Γ(x+1), where Γ(x) is the Gamma function (see Chapter 3). The factorial symbol (!) can be entered also as the keystroke combination ALPHA ↱ 2 .

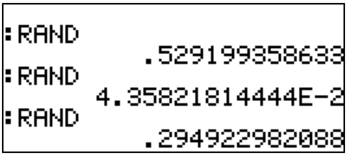
Example of applications of these functions are shown next:



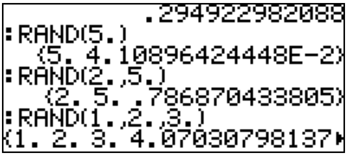
```
:COMB(10.,6.)
                    210.
:PERM(10.,6.)
                 151200.
:12.!
              479001600.
```

## Random numbers

The calculator provides a random number generator that returns a uniformly distributed, random real number between 0 and 1. The generator is able to produce sequences of random numbers. However, after a certain number of times (a very large number indeed), the sequence tends to repeat itself. For that reason, the random number generator is more properly referred to as a pseudo-random number generator. To generate a random number with your calculator use function RAND from the MTH/PROBABILITY sub-menu. The following screen shows a number of random numbers produced using RAND. The numbers in the left-hand side figure are produced with calling function RAND without an argument. If you place an argument list in function RAND, you get back the list of numbers plus an additional random number attached to it as illustrated in the right-hand side figure.



```
:RAND
           .529199358633
:RAND
     4.35821814444E-2
:RAND
           .294922982088
```



```
           .294922982088
:RAND(5.)
  {5. 4.10896424448E-2}
:RAND(2.,5.)
  {2. 5. .786870433805}
:RAND(1.,2.,3.)
{1. 2. 3. 4.07030798137▸
```

Random number generators, in general, operate by taking a value, called the "seed" of the generator, and performing some mathematical algorithm on that "seed" that generates a new (pseudo)random number. If you want to generate a sequence of number and be able to repeat the same sequence later, you can change the "seed" of the generator by using function RDZ(n), where n is the "seed," before generating the sequence. Random number generators operate by starting with a "seed" number that is transformed into the first random number of the series. The current number then serves as the "seed" for the next number and so on. By "re-seeding" the sequence with the same number you can reproduce the same sequence more than once. For example, try the following:

| | |
|---|---|
| RDZ(0.25) ENTER | Use 0.25 as the "seed." |
| RAND() ENTER | First random number = 0.75285… |
| RAND() ENTER | Second random number = 0.51109… |
| RAND() ENTER | Third random number= 0.085429…. |
| Re-start the sequence: | |
| RDZ(0.25) ENTER | Use 0.25 as the "seed." |
| RAND() ENTER | First random number = 0.75285… |
| RAND() ENTER | Second random number = 0.51109… |
| RAND() ENTER | Third random number= 0.085429…. |

To generate a sequence of random numbers use function SEQ. For example, to generate a list of 5 random numbers you can use, in ALG mode: `SEQ(RAND(),j,1,5,1)`. In RPN mode, use the following program:

```
« → n « 1 n FOR j RND NEXT n →LIST » »
```

Store it into variable RLST (Random LiST), and use VAR 5 RLST to produce a list of 5 random numbers.

Function RNDM(n,m) can be used to generate a matrix of n rows and m columns whose elements are random integers between -1 and 1(see Chapter 10).

## Discrete probability distributions

A random variable is said to be discrete when it can only take a finite number of values. For example, the number of rainy days in a given location can be considered a discrete random variable because we count them as integer numbers only. Let X represent a discrete random variable, its probability mass function (pmf) is represented by f(x) = P[X=x], i.e., the probability that the random variable X takes the value x.

The mass distribution function must satisfy the conditions that

$$f(x) > 0, \text{ for all } x,$$

and

$$\sum_{all\ x} f(x) = 1.0$$

A cumulative distribution function (cdf) is defined as

$$F(x) = P[X \le x] = \sum_{k \le x} f(k)$$

Next, we will define a number of functions to calculate discrete probability distributions. We suggest that you create a sub-directory, say, HOME\STATS\DFUN (Discrete FUNctions) where we will define the probability mass function and cumulative distribution function for the binomial and Poisson distributions.

### Binomial distribution

The probability mass function of the binomial distribution is given by

$$f(n,p,x) = \binom{n}{x} \cdot p^x \cdot (1-p)^{n-x}, \quad x = 0,1,2,\ldots,n$$

where $(^n{}_x)$ = C(n,x) is the combination of n elements taken x at a time. The values n and p are the parameters of the distribution. The value n represents the number of repetitions of an experiment or observation that can have one of two outcomes, e.g., success and failure. If the random variable X represents the number of successes in the n repetitions, then p represents the probability of getting a success in any given repetition. The cumulative distribution function for the binomial distribution is given by

$$F(n,p,x) = \sum_{k=0}^{x} f(n,p,x), \quad x = 0,1,2,...,n$$

## Poisson distribution

The probability mass function of the Poisson distribution is given by

$$f(\lambda,x) = \frac{e^{-\lambda} \cdot \lambda^{x}}{x!}, \quad x = 0,1,2,...,\infty \, .$$

In this expression, if the random variable X represents the number of occurrences of an event or observation per unit time, length, area, volume, etc., then the parameter l represents the average number of occurrences per unit time, length, area, volume, etc. The cumulative distribution function for the Poisson distribution is given by

$$F(\lambda,x) = \sum_{k=0}^{x} f(\lambda,x), \quad x = 0,1,2,...,\infty$$

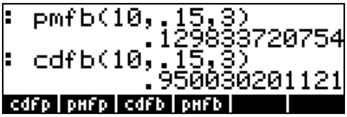
Next, use function DEFINE (↰ DEF ) to define the following probability mass functions (pmf) and cumulative distribution functions (cdf):

```
DEFINE(pmfb(n,p,x) = COMB(n,x)*p^x*(1-p)^(n-x))
DEFINE(cdfb(n,p,x) = Σ(k=0,x,pmfb(n,p,k)))
DEFINE(pmfp(λ,x) = EXP(-λ)*λ^x/x!)
DEFINE(cdfp(λ,x) = Σ(k=0,x,pmfp(λ,x)))
```

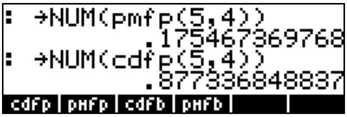
The function names stand for:

- pmfb: probability mass function for the binomial distribution
- cdfb: cumulative distribution function for the binomial distribution
- pmfp: probability mass function for the Poisson distribution
- cdfp: cumulative distribution function for the Poisson distribution

Examples of calculations using these functions are shown next:



```
: pmfb(10,.15,3)
             .129833720754
: cdfb(10,.15,3)
             .950030201121
```



```
: →NUM(pmfp(5,4))
             .175467369768
: →NUM(cdfp(5,4))
             .877336848837
```

# Continuous probability distributions

The probability distribution for a continuous random variable, X, is characterized by a function f(x) known as the probability density function (pdf). The pdf has the following properties: f(x) > 0, for all x, and

$$P[X < x] = F(x) = \int_{-\infty}^{x} f(\xi)d\xi.$$

$$\int_{-\infty}^{+\infty} f(x)dx = 1.$$

Probabilities are calculated using the cumulative distribution function (cdf), F(x), defined by $P[X < x] = F(x) = \int_{-\infty}^{x} f(\xi)d\xi$ , where P[X<x] stands for "the probability that the random variable X is less than the value x".

In this section we describe several continuous probability distributions including the gamma, exponential, beta, and Weibull distributions. These distributions are described in any statistics textbook. Some of these distributions make use of a the Gamma function defined earlier, which is calculated in the calculator by using the factorial function as Γ(x) = (x-1)!, for any real number x.

## The gamma distribution

The probability distribution function (pdf) for the gamma distribution is given by

$$f(x) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} \cdot x^{\alpha-1} \cdot \exp(-\frac{x}{\beta}), for \quad x > 0, \alpha > 0, \beta > 0;$$

The corresponding (cumulative) distribution function (cdf) would be given by an integral that has no closed-form solution.

## The exponential distribution

The exponential distribution is the gamma distribution with a = 1. Its pdf is given by

$$f(x) = \frac{1}{\beta} \cdot \exp(-\frac{x}{\beta}), for \quad x > 0, \beta > 0,$$

while its cdf is given by F(x) = 1 - exp(-x/β), for x>0, β >0.

## The beta distribution

The pdf for the gamma distribution is given by

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)} \cdot x^{\alpha-1} \cdot (1-x)^{\beta-1}, for \quad 0 < x < 1, \alpha > 0, \beta > 0$$

As in the case of the gamma distribution, the corresponding cdf for the beta distribution is also given by an integral with no closed-form solution.

## The Weibull distribution

The pdf for the Weibull distribution is given by

$$f(x) = \alpha \cdot \beta \cdot x^{\beta-1} \cdot \exp(-\alpha \cdot x^{\beta}), \quad for\ x > 0, \alpha > 0, \beta > 0$$

While the corresponding cdf is given by

$$F(x) = 1 - \exp(-\alpha \cdot x^{\beta}), \quad for\ x > 0, \alpha > 0, \beta > 0$$

## Functions for continuous distributions

To define a collection of functions corresponding to the gamma, exponential, beta, and Weibull distributions, first create a sub-directory called CFUN

(Continuous FUNctions) and define the following functions (change to Approx mode):

Gamma pdf: `'gpdf(x) = x^(α-1)*EXP(-x/β)/(β^α*GAMMA(α))'`
Gamma cdf: `'gcdf(x) = ∫(0,x,gpdf(t),t)'`
Beta pdf:
`' βpdf(x)= GAMMA(α+β)*x^(α-1)*(1-x)^(β-1)/(GAMMA(α)*GAMMA(β))'`
Beta cdf: `' βcdf(x) = ∫(0,x, βpdf(t),t)'`
Exponential pdf: `'epdf(x) = EXP(-x/β)/β'`
Exponential cdf: `'ecdf(x) = 1 - EXP(-x/β)'`
Weibull pdf: `'Wpdf(x) = α*β*x^(β-1)*EXP(-α*x^β)'`
Weibull cdf: `'Wcdf(x) = 1 - EXP(-α*x^β)'`

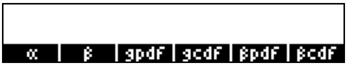
Use function DEFINE to define all these functions. Next, enter the values of α and β, e.g., 1 STO▶ ALPHA ↱ A ENTER 2 STO▶ ALPHA ↱ B ENTER

Finally, for the cdf for Gamma and Beta cdf's, you need to edit the program definitions to add →NUM to the programs produced by function DEFINE. For example, the Gamma cdf, i.e., the function gcdf, should be modified to read: « → x '→NUM( ∫ (0,x,gpdf(t),t))' » and stored back into [illegible]. Repeat the procedure for βcdf.

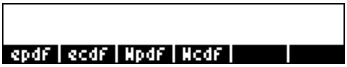
Unlike the discrete functions defined earlier, the continuous functions defined in this section do not include their parameters (α and/or β) in their definitions. Therefore, you don't need to enter them in the display to calculate the functions. However, those parameters must be previously defined by storing the corresponding values in the variables α and β. Once all functions and the values α and β have been stored, you can order the menu labels by using function ORDER. The call to the function will be the following:

ORDER({'α','β','gpdf','gcdf','βpdf','βcdf','epdf','ecdf','Wpdf','Wcdf'})

Following this command the menu labels will show as follows (Press NXT to move to the second list. Press NXT once more to move to the first list):

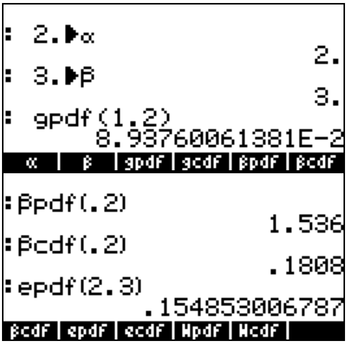


α | β | gpdf | gcdf | βpdf | βcdf



epdf | ecdf | Wpdf | Wcdf | |

Some examples of application of these functions, for values of $\alpha = 2$, $\beta = 3$, are shown below. Notice the variable IERR that shows up in the second screen shot. This results from a numerical integration for function gcdf.

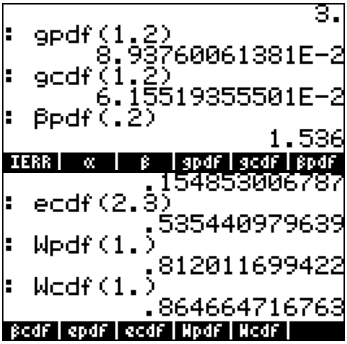


```
:  2.▶α
                       2.
:  3.▶β
                       3.
:  gpdf(1.2)
       8.93760061381E-2
```

α | β | gpdf | gcdf | βpdf | βcdf

```
:βpdf(.2)
                    1.536
:βcdf(.2)
                    .1808
:epdf(2.3)
          .154853006787
```

βcdf | epdf | ecdf | Wpdf | Wcdf |



```
                       3.
:  gpdf(1.2)
       8.93760061381E-2
:  gcdf(1.2)
       6.15519355501E-2
:  βpdf(.2)
                    1.536
```

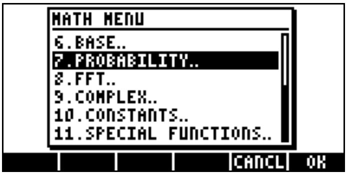
IERR | α | β | gpdf | gcdf | βpdf

```
          .154853006787
:  ecdf(2.3)
          .535440979639
:  Wpdf(1.)
          .812011699422
:  Wcdf(1.)
          .864664716763
```

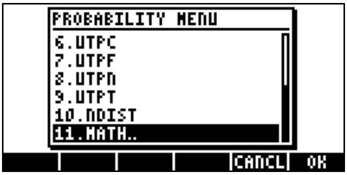
βcdf | epdf | ecdf | Wpdf | Wcdf |

## Continuous distributions for statistical inference

In this section we discuss four continuous probability distributions that are commonly used for problems related to statistical inference. These distributions are the normal distribution, the Student's t distribution, the Chi-square ($\chi^2$) distribution, and the F-distribution. The functions provided by the calculator to evaluate probabilities for these distributions are contained in the MTH/PROBABILITY menu introduced earlier in this chapter. The functions are NDIST, UTPN, UTPT, UTPC, and UTPF. Their application is described in the following sections. To see these functions activate the MTH menu: ↰ MTH and select the PROBABILITY option:



```
MATH MENU
6.BASE..
7.PROBABILITY..
8.FFT..
9.COMPLEX..
10.CONSTANTS..
11.SPECIAL FUNCTIONS..
```



```
PROBABILITY MENU
6.UTPC
7.UTPF
8.UTPN
9.UTPT
10.NDIST
11.MATH..
```

### Normal distribution pdf

The expression for the normal distribution pdf is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{(x-\mu)^2}{2\sigma^2}],$$

where μ is the mean, and $\sigma^2$ is the variance of the distribution. To calculate the value of f(μ,$\sigma^2$,x) for the normal distribution, use function NDIST with the following arguments: the mean, μ, the variance, $\sigma^2$, and, the value x , i.e., NDIST(μ,$\sigma^2$,x). For example, check that for a normal distribution, f(1.0,0.5,2.0) = 0.20755374.

## Normal distribution cdf

The calculator has a function UTPN that calculates the upper-tail normal distribution, i.e., UTPN(x) = P(X>x) = 1 - P(X<x). To obtain the value of the upper-tail normal distribution UTPN we need to enter the following values: the mean, μ; the variance, $\sigma^2$; and, the value x, e.g., UTPN((μ,$\sigma^2$,x)

For example, check that for a normal distribution, with μ = 1.0, $\sigma^2$ = 0.5, UTPN(0.75) = 0.638163. Use UTPN(1.0,0.5,0.75) = 0.638163.

Different probability calculations for normal distributions [X is N(μ,$\sigma^2$)] can be defined using the function UTPN, as follows:

- P(X<a) = 1 - UTPN(μ, $\sigma^2$,a)
- P(a<X<b) = P(X<b) - P(X<a) = 1 - UTPN(μ, $\sigma^2$,b) - (1 - UTPN(μ, $\sigma^2$,a)) = UTPN(μ, $\sigma^2$,a) - UTPN(μ, $\sigma^2$,b)
- P(X>c) = UTPN(μ, $\sigma^2$,c)

Examples: Using μ = 1.5, and $\sigma^2$ = 0.5, find:
P(X<1.0) = 1 - P(X>1.0) = 1 - UTPN(1.5, 0.5, 1.0) = 0.239750.
P(X>2.0) = UTPN(1.5, 0.5, 2.0) = 0.239750.
P(1.0<X<2.0) = F(1.0) - F(2.0) = UTPN(1.5,0.5,1.0) - UTPN(1.5,0.5,2.0) = 0.7602499 - 0.2397500 = 0.524998.

## The Student-t distribution

The Student-t, or simply, the t-, distribution has one parameter ν, known as the degrees of freedom of the distribution. The probability distribution function (pdf) is given by

$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})\cdot\sqrt{\pi\nu}} \cdot (1+\frac{t^2}{\nu})^{-\frac{\nu+1}{2}}, -\infty < t < \infty$$

where Γ(α) = (α-1)! is the GAMMA function defined in Chapter 3.

The calculator provides for values of the upper-tail (cumulative) distribution function for the t-distribution, function UTPT, given the parameter ν and the value of t, i.e., UTPT(ν,t). The definition of this function is, therefore,

$$UTPT(\nu,t) = \int_t^\infty f(t)dt = 1 - \int_{-\infty}^t f(t)dt = 1 - P(T \le t)$$

For example, UTPT(5,2.5) = 2.7245…E-2. Other probability calculations for the t-distribution can be defined using the function UTPT, as follows:

- P(T<a) = 1 - UTPT(ν,a)
- P(a<T<b) = P(T<b) - P(T<a) = 1 - UTPT(ν,b) - (1 - UTPT(ν,a)) = UTPT(ν,a) - UTPT(ν,b)
- P(T>c) = UTPT(ν,c)

Examples: Given ν = 12, determine:

P(T<0.5) = 1-UTPT(12,0.5) = 0.68694..
P(-0.5<T<0.5) = UTPT(12,-0.5)-UTPT(12,0.5) = 0.3738…
P(T> -1.2) = UTPT(12,-1.2) = 0.8733…

## The Chi-square distribution

The Chi-square ($\chi^2$) distribution has one parameter ν, known as the degrees of freedom. The probability distribution function (pdf) is given by

$$f(x)=\frac{1}{2^{\frac{\nu}{2}}\cdot\Gamma(\frac{\nu}{2})}\cdot x^{\frac{\nu}{2}-1}\cdot e^{-\frac{x}{2}},\nu>0,x>0$$

The calculator provides for values of the upper-tail (cumulative) distribution function for the $\chi^2$-distribution using [UTPC] given the value of x and the parameter ν. The definition of this function is, therefore,

$$UTPC(\nu,x)=\int_t^{\infty}f(x)dx=1-\int_{-\infty}^{t}f(x)dx=1-P(X\le x)$$

To use this function, we need the degrees of freedom, ν, and the value of the chi-square variable, x, i.e., UTPC(ν,x). For example, UTPC(5, 2.5) = 0.776495…

Different probability calculations for the Chi-squared distribution can be defined using the function UTPC, as follows:

- P(X<a) = 1 - UTPC(ν,a)
- P(a<X<b) = P(X<b) - P(X<a) = 1 - UTPC(ν,b) - (1 - UTPC(ν,a)) = UTPC(ν,a) - UTPC(ν,b)
- P(X>c) = UTPC(ν,c)

Examples: Given ν = 6, determine:

P(X<5.32) = 1-UTPC(6,5.32) = 0.4965..
P(1.2<X<10.5) = UTPC(6,1.2)-UTPC(6,10.5) = 0.8717…
P(X> 20) = UTPC(6,20) = 2.769..E-3

## The F distribution

The F distribution has two parameters νN = numerator degrees of freedom, and νD = denominator degrees of freedom. The probability distribution function (pdf) is given by

$$f(x) = \frac{\Gamma(\frac{\nu N + \nu D}{2}) \cdot (\frac{\nu N}{\nu D})^{\frac{\nu N}{2}} \cdot F^{\frac{\nu N}{2} - 1}}{\Gamma(\frac{\nu N}{2}) \cdot \Gamma(\frac{\nu D}{2}) \cdot (1 - \frac{\nu N \cdot F}{\nu D})^{(\frac{\nu N + \nu D}{2})}}$$

The calculator provides for values of the upper-tail (cumulative) distribution function for the F distribution, function UTPF, given the parameters νN and νD, and the value of F. The definition of this function is, therefore,

$$UTPF(\nu N, \nu D, F) = \int_t^\infty f(F)dF = 1 - \int_{-\infty}^t f(F)dF = 1 - P(\Im \leq F)$$

For example, to calculate UTPF(10,5, 2.5) = 0.161834…

Different probability calculations for the F distribution can be defined using the function UTPF, as follows:

- P(F<a) = 1 - UTPF(νN, νD,a)
- P(a<F<b) = P(F<b) - P(F<a) = 1 -UTPF(νN, νD,b)- (1 - UTPF(νN, νD,a))
  = UTPF(νN, νD,a) - UTPF(νN, νD,b)
- P(F>c) = UTPF(νN, νD,a)

Example: Given νN = 10, νD = 5, find:

P(F<2) = 1-UTPF(10,5,2) = 0.7700…
P(5<F<10) = UTPF(10,5,5) – UTPF(10,5,10) = 3.4693..E-2
P(F>5) = UTPF(10,5,5) = 4.4808..E-2

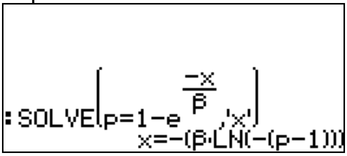
## Inverse cumulative distribution functions

For a continuous random variable X with cumulative density function (cdf) F(x) = P(X<x) = p, to calculate the inverse cumulative distribution function we need to find the value of x, such that x = $F^{-1}(p)$. This value is relatively simple to find for the cases of the exponential and Weibull distributions since their cdf's have a closed form expression:

- Exponential, F(x) = 1 - exp(-x/β)
- Weibull, F(x) = 1-exp(-α$x^{\beta}$)

(Before continuing, make sure to purge variables α and β). To find the inverse cdf's for these two distributions we need just solve for x from these expressions, i.e.,

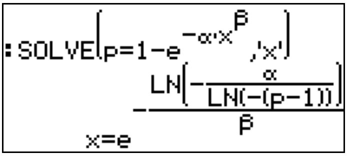
Exponential:



:SOLVE$\left(p=1-e^{\frac{-x}{\beta}},'x'\right)$

x=-(β·LN(-(p-1)))

Weibull:



:SOLVE$\left(p=1-e^{-\alpha\cdot x^{\beta}},'x'\right)$

x=$e^{-\frac{LN\left(-\frac{\alpha}{LN(-(p-1))}\right)}{\beta}}$

For the <u>Gamma and Beta distributions</u> the expressions to solve will be more complicated due to the presence of integrals, i.e.,

- Gamma, $p=\int_0^x \frac{1}{\beta^\alpha\Gamma(\alpha)}\cdot z^{\alpha-1}\cdot\exp(-\frac{z}{\beta})dz$
- Beta, $p=\int_0^x \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}\cdot z^{\alpha-1}\cdot(1-z)^{\beta-1}dz$

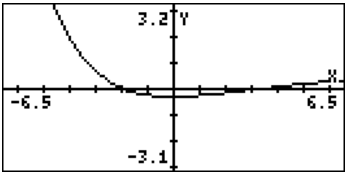
A numerical solution with the numerical solver will not be feasible because of the integral sign involved in the expression. However, a graphical solution is possible. Details on how to find the root of a graph are presented in Chapter 12. To ensure numerical results, change the CAS setting to Approx. The function to plot for the Gamma distribution is

Y(X) = ∫(0,X,z^(α-1)*exp(-z/β)/(β^α*GAMMA(α)),z)-p

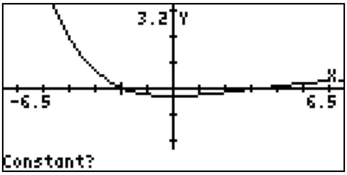
For the Beta distribution, the function to plot is

Y(X) =
∫(0,X,z^(α-1)*(1-z)^(β-1)*GAMMA(α+β)/(GAMMA(α)*GAMMA(β)),z)-p

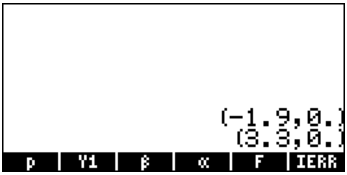
To produce the plot, it is necessary to store values of α, β, and p, before attempting the plot. For example, for α = 2, β = 3, and p = 0.3, the plot of Y(X) for the Gamma distribution is shown below. (Please notice that, because of the complicated nature of function Y(X), it will take some time before the graph is produced. Be patient.)



There are two roots of this function found by using function ROOT within the plot environment. Because of the integral in the equation, the root is approximated and will not be shown in the plot screen. You will only get the message Constant? Shown in the screen. However, if you press ENTER at this point, the approximate root will be listed in the display. Two roots are shown in the right-hand figure below.

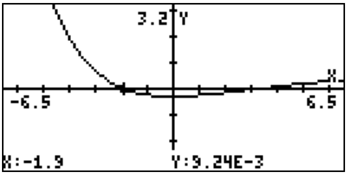


Constant?

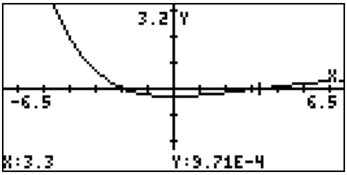


```
(-1.9,0.)
 (3.3,0.)
```

Alternatively, you can use function TRACE (X,Y) to estimate the roots by tracing the curve near its intercepts with the x-axis. Two estimates are shown below:

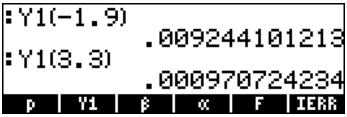


X:-1.9 Y:9.24E-3



X:3.3 Y:9.71E-4

These estimates suggest solutions x = -1.9 and x = 3.3. You can verify these "solutions" by evaluating function Y1(X) for X = -1.9 and X = 3.3, i.e.,



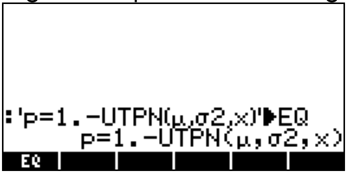
```
:Y1(-1.9)
                .009244101213
:Y1(3.3)
                .000970724234
```

For the normal, Student's t, Chi-square ($\chi^2$), and F distributions, which are represented by functions UTPN, UTPT, UPTC, and UTPF in the calculator, the inverse cuff can be found by solving one of the following equations:

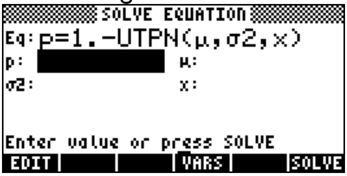
- Normal, p = 1 – UTPN(μ,σ2,x)
- Student's t, p = 1 – UTPT(ν,t)
- Chi-square, p = 1 – UTPC(ν,x)
- F distribution: p = 1 – UTPF(νN,νD,F)

Notice that the second parameter in the UTPN function is σ2, not $\sigma^2$, representing the variance of the distribution. Also, the symbol ν (the lower-case Greek letter no) is not available in the calculator. You can use, for example, γ (gamma) instead of ν. The letter γ is available thought the character set (↱ CHARS ).

For example, to obtain the value of x for a normal distribution, with μ = 10, $\sigma^2$ = 2, with p = 0.25, store the equation 'p=1-UTPN(μ,σ2,x)' into variable EQ (figure in the left-hand side below). Then, launch the numerical solver, to get the input form in the right-hand side figure:

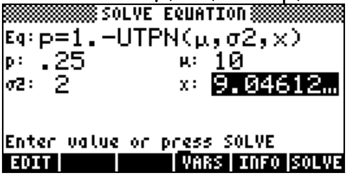


```
:'p=1.-UTPN(μ,σ2,x)'▶EQ
     p=1.-UTPN(μ,σ2,x)
```



```
SOLVE EQUATION
Eq: p=1.-UTPN(μ,σ2,x)
p:              μ:
σ2:             x:
Enter value or press SOLVE
```

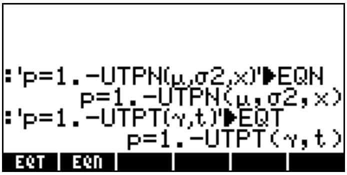
The next step is to enter the values of μ, $\sigma^2$, and p, and solve for x:



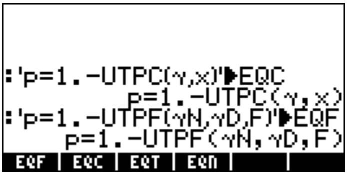
```
SOLVE EQUATION
Eq: p=1.-UTPN(μ,σ2,x)
p: .25          μ: 10
σ2: 2           x: 9.04612…
Enter value or press SOLVE
```

This input form can be used to solve for any of the four variables involved in the equation for the normal distribution.

To facilitate solution of equations involving functions UTPN, UTPT, UTPC, and UTPF, you may want to create a sub-directory UTPEQ were you will store the equations listed above:

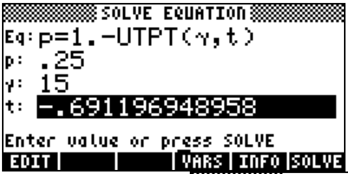


```
:'p=1.-UTPN(μ,σ2,x)'▶EQN
                p=1.-UTPN(μ,σ2,x)
:'p=1.-UTPT(γ,t)'▶EQT
                   p=1.-UTPT(γ,t)
```

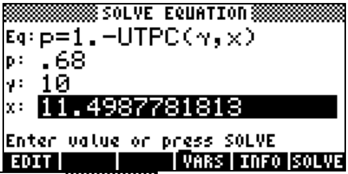


```
:'p=1.-UTPC(γ,x)'▶EQC
                  p=1.-UTPC(γ,x)
:'p=1.-UTPF(γN,γD,F)'▶EQF
              p=1.-UTPF(γN,γD,F)
```

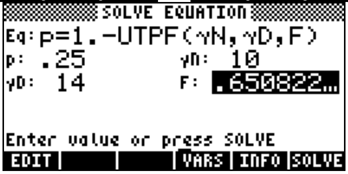
Thus, at this point, you will have the four equations available for solution. You needs just load one of the equations into the EQ field in the numerical solver and proceed with solving for one of the variables. Examples of the UTPT, UTPC, and UPTF are shown below:



```
SOLVE EQUATION
Eq: p=1.-UTPT(γ,t)
p: .25
γ: 15
t: -.691196948958
Enter value or press SOLVE
```

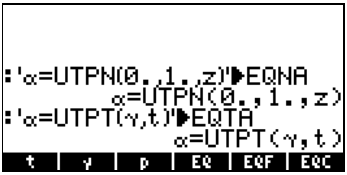


```
SOLVE EQUATION
Eq: p=1.-UTPC(γ,x)
p: .68
γ: 10
x: 11.4987781813
Enter value or press SOLVE
```

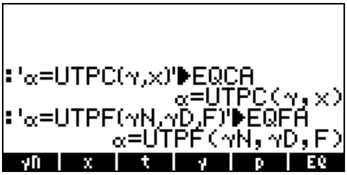


```
SOLVE EQUATION
Eq: p=1.-UTPF(γN,γD,F)
p: .25          γN: 10
γD: 14          F: .650822…
Enter value or press SOLVE
```

Notice that in all the examples shown above, we are working with p = P(X<x). In many statistical inference problems we will actually try to find the value of x for which P(X>x) = $\alpha$. Furthermore, for the normal distribution, we most likely will be working with the standard normal distribution in which $\mu$ =0, and $\sigma^2$ = 1. The standard normal variable is typically referred to as Z, so that the problem to solve will be P(Z>z) = $\alpha$. For these cases of statistical inference problems, we could store the following equations:

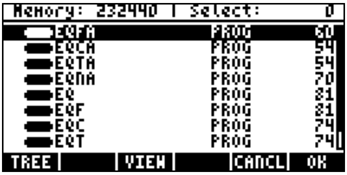


```
:'α=UTPN(0.,1.,z)'▶EQNA
              α=UTPN(0.,1.,z)
:'α=UTPT(γ,t)'▶EQTA
                  α=UTPT(γ,t)
```



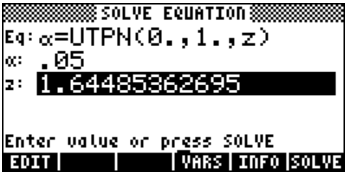
```
:'α=UTPC(γ,x)'▶EQCA
                  α=UTPC(γ,x)
:'α=UTPF(γN,γD,F)'▶EQFA
              α=UTPF(γN,γD,F)
```

With these four equations, whenever you launch the numerical solver you have the following choices:

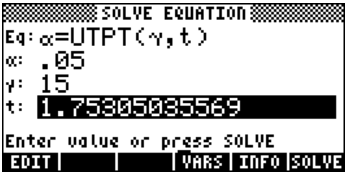


```
Memory: 232440 | Select:      0
     EQFA           PROG     60
     EQCA           PROG     54
     EQTA           PROG     54
     EQNA           PROG     70
     EQ             PROG     81
     EQF            PROG     81
     EQC            PROG     74
     EQT            PROG     74
```

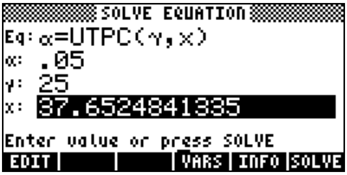
Examples of solution of equations EQNA, EQTA, EQCA, and EQFA are shown below:



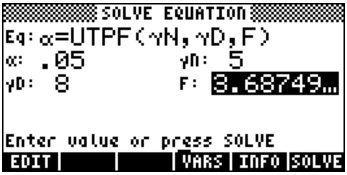
```
SOLVE EQUATION
Eq: α=UTPN(0.,1.,z)
α:  .05
z:  1.64485362695

Enter value or press SOLVE
```



```
SOLVE EQUATION
Eq: α=UTPT(γ,t)
α:  .05
γ:  15
t:  1.75305035569

Enter value or press SOLVE
```



```
SOLVE EQUATION
Eq: α=UTPC(γ,x)
α:  .05
γ:  25
x:  37.6524841335

Enter value or press SOLVE
```



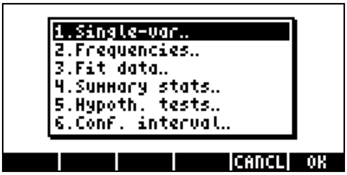
```
SOLVE EQUATION
Eq: α=UTPF(γN,γD,F)
α:  .05          γN:  5
γD:  8           F:  3.68749…

Enter value or press SOLVE
```

# Chapter 18
# Statistical Applications

In this Chapter we introduce statistical applications of the calculator including statistics of a sample, frequency distribution of data, simple regression, confidence intervals, and hypothesis testing.

## Pre-programmed statistical features

The calculator provides pre-programmed statistical features that are accessible through the keystroke combination ↱ STAT (same key as the number 5 key). The statistical applications available in the calculator are:



1.Single-var..
2.Frequencies..
3.Fit data..
4.Summary stats..
5.Hypoth. tests..
6.Conf. interval..

These applications are presented in detail in this Chapter. First, however, we demonstrate how to enter data for statistical analysis.

### Entering data

For the analysis of a single set of data (a sample) we can use applications number 1, 2, and 4 from the list above. All of these applications require that the data be available as columns of the matrix ΣDAT. This can be accomplished by entering the data in columns using the matrix writer, ↰ MTRW .

This operation may become tedious for large number of data points. Instead, you may want to enter the data as a list (see Chapter 8) and convert the list into a column vector by using program CRMC (see Chapter 10). Alternatively, you can enter the following program to convert a list into a column vector. Type the program in RPN mode:

```
« OBJ→ 1 2 →LIST →ARRY »
```

Store the program in a variable called LXC. After storing this program in RPN mode you can also use it in ALG mode.

To store a column vector into variable ΣDAT use function STOΣ, available through the catalog (↱ CAT ), e.g., STOΣ (ANS(1)) in ALG mode.

Example 1 – Using the program LXC, defined above, create a column vector using the following data: 2.1 1.2 3.1 4.5 2.3 1.1 2.3 1.5 1.6 2.2 1.2 2.5.

In RPG mode, type in the data in a list:

{2.1 1.2 3.1 4.5 2.3 1.1 2.3 1.5 1.6 2.2 1.2 2.5 } ENTER LXC

Use function STOΣ to store the data into ΣDAT.

## Calculating single-variable statistics

Assuming that the single data set was stored as a column vector in variable ΣDAT. To access the different STAT programs, press ↱ STAT . Press OK to select **1. Single-var..** There will be available to you an input form labeled **SINGLE-VARIABLE STATISTICS**, with the data currently in your ΣDAT variable listed in the form as a vector. Since you only have one column, the field **Col:** should have the value 1 in front of it. The **Type** field determines whether you are working with a sample or a population, the default setting is Sample. Move the cursor to the horizontal line preceding the fields **Mean**, **Std Dev**, **Variance**, **Total**, **Maximum**, **Minimum**, pressing the ✓CHK soft menu key to select those measures that you want as output of this program. When ready, press OK. The selected values will be listed, appropriately labeled, in the screen of your calculator.

Example 1 -- For the data stored in the previous example, the single-variable statistics results are the following:

Mean: 2.133, Std Dev: 0.964, Variance: 0.929
Total: 25.6, Maximum: 4.5, Minimum: 1.1

## Definitions

The definitions used for these quantities are the following:

Suppose that you have a number data points $x_1$, $x_2$, $x_3$, …, representing different measurements of the same discrete or continuous variable x. The set of all possible values of the quantity x is referred to as the population of x. A finite population will have only a fixed number of elements $x_i$. If the quantity x represents the measurement of a continuous quantity, and since, in theory, such a quantity can take an infinite number of values, the population of x in this case is infinite. If you select a sub-set of a population, represented by the n data values {$x_1$, $x_2$, …, $x_n$}, we say you have selected a sample of values of x.

Samples are characterized by a number of measures or statistics. There are measures of central tendency, such as the mean, median, and mode, and measures of spreading, such as the range, variance, and standard deviation.

### *Measures of central tendency*

The mean (or arithmetic mean) of the sample, $\bar{x}$, is defined as the average value of the sample elements,

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i .$$

The value labeled `Total` obtained above represents the summation of the values of x, or $\Sigma x_i = n \cdot \bar{x}$. This is the value provided by the calculator under the heading `Mean`. Other mean values used in certain applications are the geometric mean, $x_g$, or the harmonic mean, $x_h$, defined as:

$$x_g = \sqrt[n]{x_1 \cdot x_2 \cdots x_n}, \qquad \frac{1}{x_h} = \sum_{i=1}^{n} \frac{1}{x_i}.$$

Examples of calculation of these measures, using lists, are available in Chapter 8.

The median is the value that splits the data set in the middle when the elements are placed in increasing order. If you have an odd number, n, of ordered elements, the median of this sample is the value located in position (n+1)/2. If you have an even number, n, of elements, the median is the average of the elements located in positions n/2 and (n+1)/2. Although the pre-programmed statistical features of the calculator do not include the calculation of the median, it is very easily to write a program to calculate such quantity by working with lists. For example, if you want to use the data in ΣDAT to find the median, type the following program in RPN mode (see Chapter 21 for more information on programming in User RPL language).:

```
« → nC « RCLΣ DUP SIZE 2 GET IF 1 > THEN nC COL– SWAP DROP
OBJ→ 1 + →ARRY END OBJ→ OBJ→ DROP DROP DUP → n « →LIST SORT
IF ‘n MOD 2 == 0’ THEN DUP ‘n/2’ EVAL GET SWAP ‘(n+1)/2’ EVAL GET +
2 / ELSE ‘(n+1)/2’ EVAL GET END “Median” →TAG » » »
```

Store this program under the name MED. An example of application of this program is shown next.

Example 2 – To run the program, first you need to prepare your ΣDAT matrix. Then, enter the number of the column in ΣDAT whose median you want to find, and press MED. For the data currently in ΣDAT (entered in an earlier example), use program MED to show that `Median: 2.15`.

The mode of a sample is better determined from histograms, therefore, we leave its definition for a later section.

### *Measures of spread*

The variance (Var) of the sample is defined as $s_x^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^2$ .

The standard deviation (St Dev) of the sample is just the square root of the variance, i.e., $s_x$.

The range of the sample is the difference between the maximum and minimum values of the sample. Since the calculator, through the pre-programmed statistical functions provides the maximum and minimum values of the sample, you can easily calculate the range.

***Coefficient of variation***

The coefficient of variation of a sample combines the mean, a measure of central tendency, with the standard deviation, a measure of spreading, and is defined, as a percentage, by: $V_x = (s_x/ \bar{x})100$.

<u>Sample vs. population</u>

The pre-programmed functions for single-variable statistics used above can be applied to a finite population by selecting the `Type: Population` in the `SINGLE-VARIABLE STATISTICS` screen. The main difference is in the values of the variance and standard deviation which are calculated using n in the denominator of the variance, rather than (n-1).

<u>Example 3</u> -- If you were to repeat the exercise in Example 1 of this section, using `Population` rather than `Sample` as the `Type`, you will get the same values for the mean, total, maximum, and minimum. The variance and standard deviation, however, will be given by: Variance: 0.852, Std Dev: 0.923.

## Obtaining frequency distributions

The application **2. Frequencies..** in the STAT menu can be used to obtain frequency distributions for a set of data. Again, the data must be present in the form of a column vector stored in variable ΣDAT. To get started, press ↱ STAT ▼ OK. The resulting input form contains the following fields:

**ΣDAT**: the matrix containing the data of interest.
**Col**: the column of ΣDAT that is under scrutiny.
**X-Min**: the minimum class boundary (default = -6.5).
**Bin Count**: the number of classes(default = 13).
**Bin Width**: the uniform width of each class (default = 1).

### Definitions

To understand the meaning of these parameters we present the following <u>definitions</u>: Given a set of n data values: $\{x_1, x_2, \ldots, x_n\}$ listed in no particular order, it is often required to group these data into a series of <u>classes</u> by counting the <u>frequency</u> or number of values corresponding to each class. (Note: the calculators refers to classes as bins).

Suppose that the classes, or bins, will be selected by dividing the interval ($x_{bot}$, $x_{top}$), into k = Bin Count classes by selecting a number of class boundaries, i.e., {$xB_1$, $xB_2$, …, $xB_{k+1}$}, so that class number 1 is limited by $xB_1$-$xB_2$, class number 2 by $xB_2$- $xB_3$, and so on. The last class, class number k, will be limited by $xB_k$ - $xB_{k+1}$.

The value of x corresponding to the middle of each class is known as the class mark, and is defined as $xM_i = (xB_i + xB_{i+1})/2$, for i = 1, 2, …, k.

If the classes are chosen such that the class size is the same, then we can define the class size as the value Bin Width = $\Delta x = (x_{max} - x_{min}) / k$,

and the class boundaries can be calculated as $xB_i = x_{bot} + (i - 1) * \Delta x$.

Any data point, $x_j$, j = 1, 2, …, n, belongs to the i-th class, if $xB_i \leq x_j < xB_{i+1}$

The application **2. Frequencies..** in the STAT menu will perform this frequency count, and will keep track of those values that may be below the minimum and above the maximum class boundaries (i.e., the outliers).

Example 1 -- In order to better illustrate obtaining frequency distributions, we want to generate a relatively large data set, say 200 points, by using the following:

- First, seed the random number generator using: RDZ(25) in ALG mode, or 25 ENTER RDZ in RPN mode (see Chapter 17).
- Type in the following program in RPN mode:
  « → n « 1 n FOR j RAND 100 * 2 RND NEXT n →LIST » »
  and save it under the name RDLIST (RanDom number LIST generator).
- Generate the list of 200 number by using RDLIST(200) in ALG mode, or 200 ENTER RDLIS in RPN mode.
- Use program LXC (see above) to convert the list thus generated into a column vector.
- Store the column vector into ΣDAT, by using function STOΣ.

- Obtain single-variable information using: ↱ STAT OK. Use Sample for the Type of data set, and select all options as results. The results for this example were:

Mean: 51.0406, Std Dev: .29.5893…, Variance: 875.529…
Total: 10208.12, Maximum: 99.35, Minimum: 0.13

This information indicates that our data ranges from values close to zero to values close to 100. Working with whole numbers, we can select the range of variation of the data as (0,100). To produce a frequency distribution we will use the interval (10,90) dividing it into 8 bins of width 10 each.

- Select the program **2. Frequencies..** by using ↱ STAT ▼ OK. The data is already loaded in ΣDAT, and the option Col should hold the value 1 since we have only one column in ΣDAT.
- Change X-Min to 10, Bin Count to 8, and Bin Width to 10, then press OK.

Using the RPN mode, the results are shown in the stack as a column vector in stack level 2, and a row vector of two components in stack level 1. The vector in stack level 1 is the number of outliers outside of the interval where the frequency count was performed. For this case, I get the values [ 25. 22.] indicating that there are, in my ΣDAT vector, 25 values smaller than 10 and 22 larger than 90.

- Press ◄ to drop the vector of outliers from the stack. The remaining result is the frequency count of data. This can be translated into a table as shown below.

This table was prepared from the information we provided to generate the frequency distribution, although the only column returned by the calculator is the Frequency column ($f_i$). The class numbers, and class boundaries are easy to calculate for uniform-size classes (or bins), and the class mark is just the average of the class boundaries for each class. Finally, the cumulative frequency is obtained by adding to each value in the last column, except the first, the frequency in the next row, and replacing the result in the last column of the next row. Thus, for the second class, the cumulative frequency is 18+15 = 33, while for class number 3, the cumulative frequency is 33 + 16 = 49, and so on. The cumulative frequency represents the frequency of those numbers that are smaller than or equal to the upper boundary of any given class.

| Class No. | Class | Bound. | Class mark. | Frequency | Cumulative |
|---|---|---|---|---|---|
| i | $XB_i$ | $XB_{i+1}$ | $Xm_i$ | $f_i$ | frequency |
| $< XB_1$ | outlier | below range | | 25 | |
| 1 | 10 | 20 | 15 | 18 | 18 |
| 2 | 20 | 30 | 25 | 14 | 32 |
| 3 | 30 | 40 | 35 | 17 | 49 |
| 4 | 40 | 50 | 45 | 17 | 66 |
| 5 | 50 | 60 | 55 | 22 | 88 |
| 6 | 60 | 70 | 65 | 22 | 110 |
| 7 | 70 | 80 | 75 | 24 | 134 |
| k = 8 | 80 | 90 | 85 | 19 | 153 |
| $>XB_k$ | outliers | above range | | 22 | |

Given the (column) vector of frequencies generated by the calculator, you can obtain a cumulative frequency vector by using the following program in RPN mode:

« DUP SIZE 1 GET → freq k « {k 1} 0 CON → cfreq « 'freq(1,1)' EVAL 'cfreq(1,1)' STO 2 k FOR j 'cfreq(j-1,1) +freq(j,1)' EVAL 'cfreq (j,1)' STO NEXT cfreq » » »

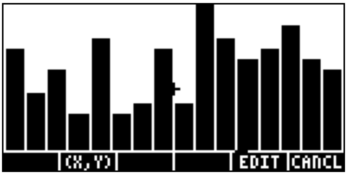
Save it under the name CFREQ. Use this program to generate the list of cumulative frequencies (press CFREQ with the column vector of frequencies in the stack). The result, for this example, is a column vector representing the last column of the table above.

### Histograms

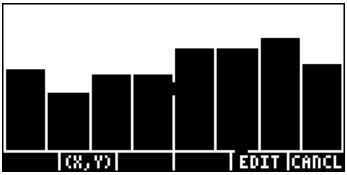
A histogram is a bar plot showing the frequency count as the height of the bars while the class boundaries shown the base of the bars. If you have your raw data (i.e., the original data before the frequency count is made) in the variable ΣDAT, you can select `Histogram` as your graph type and provide information regarding the initial value of x, the number of bins, and the bin width, to generate the histogram. Alternatively, you can generate the column vector containing the frequency count, as performed in the example above, store this vector into ΣDAT, and select `Barplot` as your graph type. In the next example, we show you how to use the first method to generate a histogram.

Example 1 – Using the 200 data points generated in the example above (stored as a column vector in ΣDAT), generate a histogram plot of the data using X-Min = 10, Bin Count = 16, and Bin Width = 5.

- First, press ↰ 2D/3D (simultaneously, if in RPN mode) to enter the PLOT SETUP screen. Within this screen, change Type: to Histogram, and check that the option Col: 1 is selected. Then, press NXT OK.
- Next, press ↰ WIN (simultaneously, if in RPN mode) to enter the PLOT WINDOW – HISTOGRAM screen. Within that screen modify the information to H-View: 10  90, V-View: 0  15, Bar Width: 5.
- Press ERASE DRAW to generate the following histogram:



- Press CANCL to return to the previous screen. Change the V-view and Bar Width once more, now to read V-View: 0  30, Bar Width: 10. The new histogram, based on the same data set, now looks like this:



A plot of frequency count, $f_i$, vs. class marks, $xM_i$, is known as a frequency polygon. A plot of the cumulative frequency vs. the upper boundaries is known as a cumulative frequency ogive. You can produce scatterplots that simulate these two plots by entering the proper data in columns 1 and 2 of a new ΣDAT matrix and changing the `Type`: to `SCATTER` in the PLOT SETUP window.

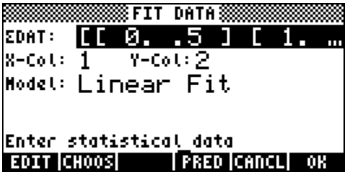
## Fitting data to a function y = f(x)

The program **3. Fit data..**, available as option number 3 in the STAT menu, can be used to fit linear, logarithmic, exponential, and power functions to data sets (x,y), stored in columns of the ΣDAT matrix. In order for this program to be effective, you need to have at least two columns in your ΣDAT variable.

<u>Example 1</u> – Fit a linear relationship to the data shown in the table below:

| **x** | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| **y** | 0.5 | 2.3 | 3.6 | 6.7 | 7.2 | 11 |

- First, enter the two rows of data into column in the variable ΣDAT by using the matrix writer, and function STOΣ.
- To access the program **3. Fit data..**, use the following keystrokes: ↱ STAT ▽ ▽ OK The input form will show the current ΣDAT, already loaded. If needed, change your set up screen to the following parameters for a linear fitting:


FIT DATA
ΣDAT: [[ 0. .5 ] [ 1. …
X-Col: 1 Y-Col: 2
Model: Linear Fit
Enter statistical data

- To obtain the data fitting press OK. The output from this program, shown below for our particular data set, consists of the following three lines in RPN mode:

```
3: '0.195238095238 + 2.00857142857*X'
2: Correlation: 0.983781424465
1: Covariance: 7.03
```

Level 3 shows the form of the equation. In this case, y = 0.06924 + 0.00383 x. Level 2 shows the sample correlation coefficient, and level 1 shows the covariance of x-y.

**Definitions**

For a sample of data points (x,y), we define the sample covariance as

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

The sample correlation coefficient for x,y is defined as

$$r_{xy} = \frac{s_{xy}}{s_x \cdot s_y}.$$

Where $s_x$, $s_y$ are the standard deviations of x and y, respectively, i.e.

$$s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad s_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

The values $s_{xy}$ and $r_{xy}$ are the "Covariance" and "Correlation," respectively, obtained by using the "Fit data" feature of the calculator.

**Linearized relationships**

Many curvilinear relationships "straighten out" to a linear form. For example, the different models for data fitting provided by the calculator can be linearized as described in the table below.

| Type of Fitting | Actual Model | Linearized Model | Indep. variable $\xi$ | Depend. Variable $\eta$ | Covar. $s_{\xi\eta}$ |
|---|---|---|---|---|---|
| Linear | y = a + bx | [same] | x | y | $s_{xy}$ |
| Log. | y = a + b ln(x) | [same] | ln(x) | y | $s_{\ln(x),y}$ |
| Exp. | $y = a e^{bx}$ | ln(y) = ln(a) + bx | x | ln(y) | $s_{x,\ln(y)}$ |
| Power | $y = a x^b$ | ln(y) = ln(a) + b ln(x) | ln(x) | ln(y) | $s_{\ln(x),\ln(y)}$ |

The sample covariance of ξ,η is given by $s_{\xi\eta} = \frac{1}{n-1}\sum(\xi_i - \overline{\xi})(\eta_i - \overline{\eta})$

Also, we define the sample variances of ξ and η, respectively, as

$$s_\xi^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\xi_i - \overline{\xi})^2 \qquad s_\eta^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\eta_i - \overline{\eta})^2$$

The sample correlation coefficient $r_{\xi\eta}$ is $r_{\xi\eta} = \frac{s_{\xi\eta}}{s_\xi \cdot s_\eta}$

The general form of the regression equation is η = A + Bξ.

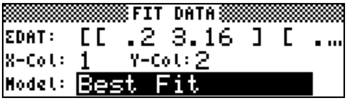
### Best data fitting

The calculator can determine which one of its linear or linearized relationship offers the best fitting for a set of (x,y) data points. We will illustrate the use of this feature with an example. Suppose you want to find which one of the data fitting functions provides the best fit for the following data:

| x | 0.2 | 0.5 | 1 | 1.5 | 2 | 4 | 5 | 10 |
|---|---|---|---|---|---|---|---|---|
| y | 3.16 | 2.73 | 2.12 | 1.65 | 1.29 | 0.47 | 0.29 | 0.01 |

First, enter the data as a matrix, either by using the matrix editor and entering the data, or by entering two lists of data corresponding to x and y and using the program CRMC developed in Chapter 10. Next, save this matrix into the statistical matrix ΣDAT, by using function STOΣ.

Finally, launch the data fit application by using: ↱ STAT ▼ ▼ OK . The display shows the current ΣDAT, already loaded. Change your set up screen to the following parameters if needed:



```
FIT DATA
ΣDAT: [[ .2 3.16 ] [ ...
X-Col: 1   Y-Col: 2
Model: Best Fit
```

Press OK, to get:

```
1: '3.99504833324*EXP(-.579206831203*X)'
2: Correlation: -0.996624999526
3: Covariance: -6.23350666124
```

The best fit for the data is, therefore, y = 3.995 $e^{-0.58 \cdot x}$.

## Obtaining additional summary statistics

The application **4. Summary stats..** in the STAT menu can be useful in some calculations for sample statistics. To get started, press ↱ STAT once more, move to the fourth option using the down-arrow key ▼, and press OK. The resulting input form contains the following fields:

**ΣDAT:** the matrix containing the data of interest.

**X-Col, Y-Col:** these options apply only when you have more than two columns in the matrix ΣDAT. By default, the x column is column 1, and the y column is column 2.

_Σ**X** _ Σ**Y...**: summary statistics that you can choose as results of this program by checking the appropriate field using [✓CHK] when that field is selected.

Many of these summary statistics are used to calculate statistics of two variables (x,y) that may be related by a function y = f(x). Therefore, this program can be thought off as a companion to program **3. Fit data..**

<u>Example 1</u> – For the x-y data currently in ΣDAT, obtain all the summary statistics.

- To access the **summary stats...** option, use: ↱ STAT ▼ ▼ ▼ OK
- Select the column numbers corresponding to the x- and y-data, i.e., X-Col: 1, and Y-Col: 2.
- Using the ✓CHK key select all the options for outputs, i.e., _ΣX, _ΣY, etc.
- Press OK to obtain the following results:

ΣX: 24.2, ΣY: 11.72, ΣX2: 148.54, ΣY2: 26.6246, ΣXY: 12.602, NΣ:8

**Note:** There are two other applications under the STAT menu, namely, **5. Hypth. tests..** and **6. Conf. Interval..** These two applications will be discussed later in the chapter.

## Calculation of percentiles

Percentiles are measures that divide a data set into 100 parts. The basic procedure to calculate the 100·p-th Percentile (0 < p < 1) in a sample of size n is as follows:

1. Order the n observations from smallest to largest.
2. Determine the product n·p
   A. If n·p is not an integer, round it up to the next integer and find the corresponding ordered value.
   B. If n·p is an integer, say k, calculate the mean of the k-th and (k-1) th ordered observations.

**Note**: Integer rounding rule, for a non-integer x.yz…, if y ≥ 5, round up to x+1; if y < 5, round up to x.

This algorithm can be implemented in the following program typed in RPN mode (See Chapter 21 for programming information):

```
« SORT DUP SIZE → p X n « n p * → k « IF k CEIL k FLOOR - NOT THEN X
k GET X k 1 + GET + 2 / ELSE k 0 RND X SWAP GET END » » »
```

which we'll store in variable %TILE (percent-tile). This program requires as input a value p within 0 and 1, representing the 100p percentile, and a list of values. The program returns the 100p percentile of the list.

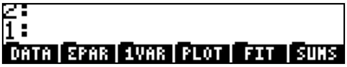
Example 1 - Determine the 37% percentile of the list { 2 1 0 1 3 5 1 2 3 6 7 9}. In RPN mode, enter 0.27 ENTER { 2 1 0 1 3 5 1 2 3 6 7 9} ENTER %TILE. In ALG mode, enter %TILE(0.27,{2,1,0,1,3,5,1,2,3,6,7,9}. The result is 1.

## The STAT soft menu

All the pre-programmed statistical functions described above are accessible through a STAT soft menu. The STAT soft menu can be accessed by using, in RPN mode, the command: 96 MENU

You can create your own program, say STATm, to activate the STAT soft menu directly. The contents of this program are simply: « 96 MENU ».

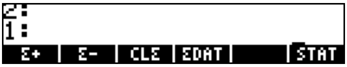
The STAT soft menu contains the following functions:



Pressing the key corresponding to any of these menus provides access to different functions as described below.

### The DATA sub-menu

The DATA sub-menu contains functions used to manipulate the statistics matrix ΣDATA:



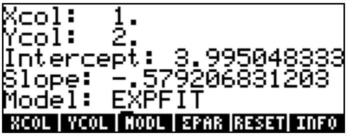
The operation of these functions is as follows:

Σ+ : add row in level 1 to bottom of ΣDATA matrix.
Σ- : removes last row in ΣDATA matrix and places it in level of 1 of the stack. The modified ΣDATA matrix remains in memory.
CLΣ : erases current ΣDATA matrix.

ΣDAT: places contents of current ΣDATA matrix in level 1 of the stack.
↰ΣDAT: stores matrix in level 1 of stack into ΣDATA matrix.

## The ΣPAR sub-menu

The ΣPAR sub-menu contains functions used to modify statistical parameters. The parameters shown correspond to the last example of data fitting.



```
Xcol:  1.
Ycol:  2.
Intercept: 3.995048333
Slope: -.579206831203
Model: EXPFIT
```

The parameters shown in the display are:

Xcol: indicates column of ΣDATA representing x (Default: 1)
Ycol: indicates column of ΣDATA representing y (Default: 2)
Intercept: shows intercept of most recent data fitting ((Default: 0)
Slope: shows slope of most recent data fitting (Default: 0)
Model: shows current data fit model (Default: LINFIT)

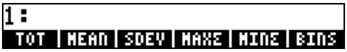
The functions listed in the soft menu keys operate as follows:
XCOL: entered as n XCOL, changes Xcol to n.
YCOL: entered as n YCOL, changes Ycol to n.
ΣPAR: shows statistical parameters.
RESET: reset parameters to default values
INFO: shows statistical parameters

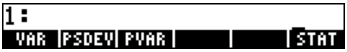
### The MODL sub-menu within ΣPAR

This sub-menu contains functions that let you change the data-fitting model to LINFIT, LOGFIT, EXPFIT, PWRFIT or BESTFIT by pressing the appropriate button.

## The 1VAR sub menu

The 1VAR sub menu contains functions that are used to calculate statistics of columns in the ΣDATA matrix.



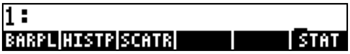


The functions available are the following:

TOT: show sum of each column in ΣDATA matrix.
MEAN: shows average of each column in ΣDATA matrix.
SDEV: shows standard deviation of each column in ΣDATA matrix.
MAXΣ: shows maximum value of each column in ΣDATA matrix.
MINΣ: shows average of each column in ΣDATA matrix.
BINS: used as $x_s$, $\Delta x$, n [BINS], provides frequency distribution for data in Xcol column in ΣDATA matrix with the frequency bins defined as [$x_s$,$x_s+\Delta x$], [$x_s$,$x_s+2\Delta x$],…, [$x_s$,$x_s+n\Delta x$].
VAR: shows variance of each column in ΣDATA matrix.
PSDEV: shows population standard deviation (based on n rather than on (n-1)) of each column in ΣDATA matrix.
PVAR: shows population variance of each column in ΣDATA matrix.
MINΣ: shows average of each column in ΣDATA matrix.

## The PLOT sub-menu

The PLOT sub-menu contains functions that are used to produce plots with the data in the ΣDATA matrix.

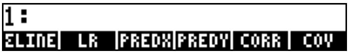


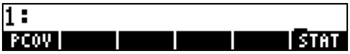
The functions included are:

BARPL: produces a bar plot with data in Xcol column of the ΣDATA matrix.
HISTP: produces histogram of the data in Xcol column in the ΣDATA matrix, using the default width corresponding to 13 bins unless the bin size is modified using function BINS in the 1VAR sub-menu (see above).
SCATR: produces a scatterplot of the data in Ycol column of the ΣDATA matrix vs. the data in Xcol column of the ΣDATA matrix. Equation fitted will be stored in the variable EQ.

## The FIT sub-menu

The FIT sub-menu contains functions used to fit equations to the data in columns Xcol and Ycol of the ΣDATA matrix.



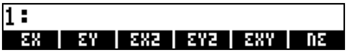


The functions available in this sub-menu are:

ΣLINE: provides the equation corresponding to the most recent fitting.
LR: provides intercept and slope of most recent fitting.
PREDX: used as y PREDX, given y find x for the fitting y = f(x).
PREDY: used as x PREDY, given x find y for the fitting y = f(x).
CORR: provides the correlation coefficient for the most recent fitting.
COV: provides sample co-variance for the most recent fitting
PCOV: shows population co-variance for the most recent fitting.

## The SUMS sub-menu

The SUMS sub-menu contains functions used to obtain summary statistics of the data in columns Xcol and of the ΣDATA matrix.



ΣX : provides the sum of values in Xcol column.
ΣY : provides the sum of values in Ycol column.
ΣX^2 : provides the sum of squares of values in Xcol column.
ΣY^2 : provides the sum of squares of values in Ycol column.
ΣX*Y: provides the sum of x·y, i.e., the products of data in columns Xcol and Ycol.
NΣ : provides the number of columns in the ΣDATA matrix.

## Example of STAT menu operations

Let ΣDATA be the matrix shown in next page.

- Type the matrix in level 1 of the stack by using the matrix editor.
- To store the matrix into ΣDATA, use: DATA ↰ ΣDAT

- Calculate statistics of each column: STAT ΣVAR:

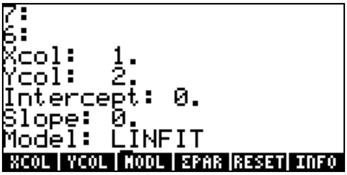
| | |
|---|---|
| TOT | produces [38.5 87.5 82799.8] |
| MEAN | produces [5.5. 12.5 11828.54…] |
| SDEV | produces [3.39… 6.78… 21097.01…] |
| MAXΣ | produces [10 21.5 55066] |
| MINΣ | produces [1.1 3.7  7.8] |
| NXT VAR | produces [11.52 46.08 445084146.33] |
| PSDEV | produces [3.142…  6.284… 19532.04…] |
| PVAR | produces [9.87…   39.49…   381500696.85…] |

  - Data:

$$\begin{bmatrix} 1.1 & 3.7 & 7.8 \\ 3.7 & 8.9 & 101 \\ 2.2 & 5.9 & 25 \\ 5.5 & 12.5 & 612 \\ 6.8 & 15.1 & 2245 \\ 9.2 & 19.9 & 24743 \\ 10.0 & 21.5 & 55066 \end{bmatrix}$$

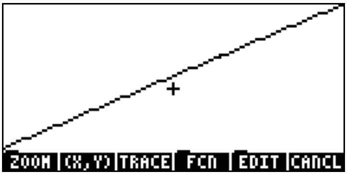
- Generate a scatterplot of the data in columns 1 and 2 and fit a straight line to it:

| | |
|---|---|
| STAT ΣPAR RESET | resets statistical parameters |



```
7:
6:
Xcol:   1.
Ycol:   2.
Intercept: 0.
Slope: 0.
Model: LINFIT
```

| | |
|---|---|
| NXT STAT PLOT SCATR | produces scatterplot |
| STATL | draws data fit as a straight line |



| | |
|---|---|
| CANCL | returns to main display |

- Determine the fitting equation and some of its statistics:

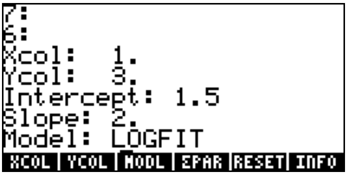
| | |
|---|---|
| STAT FIT ΣLINE | produces `'1.5+2*X'` |
| LR | produces `Intercept: 1.5, Slope: 2` |
| 3 PREDX | produces 0.75 |
| 1 PREDY | produces 3. 50 |
| CORR | produces 1.0 |
| COV | produces 23.04 |
| NXT PCOV | produces 19.74… |

- Obtain summary statistics for data in columns 1 and 2: STAT SUMS:

| | |
|---|---|
| ΣX | produces 38.5 |
| ΣY | produces 87.5 |
| ΣX2 | produces 280.87 |
| ΣY2 | produces 1370.23 |
| ΣXY | produces 619.49 |
| NΣ | produces 7 |

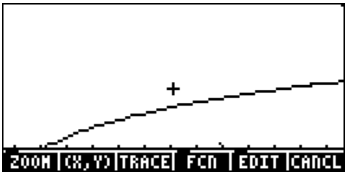
- Fit data using columns 1 (x) and 3 (y) using a logarithmic fitting:

| | |
|---|---|
| NXT STAT ΣPAR 3 YCOL | select Ycol = 3, and |
| MODL LOGFI | select Model = Logfit |



```
7:
6:
Xcol:  1.
Ycol:  3.
Intercept: 1.5
Slope: 2.
Model: LOGFIT
```

| | |
|---|---|
| NXT STAT PLOT SCATR | produce scattergram of y vs. x |
| STATL | show line for log fitting |

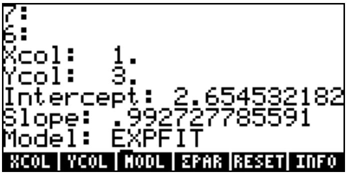


Obviously, the log-fit is not a good choice.

| | |
|---|---|
| CANCL | returns to normal display. |

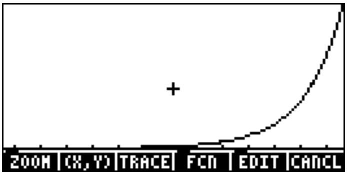
- Select the best fitting by using:

| | |
|---|---|
| STAT ΣPAR MODL BESTF | shows EXPFIT as the best fit for these data |



```
7:
6:
Xcol:  1.
Ycol:  3.
Intercept: 2.654532182
Slope: .992727785591
Model: EXPFIT
```

| | |
|---|---|
| NXT STAT FIT ΣLINE | produces '2.6545*EXP(0.9927*X)' |
| CORR | produces 0.99995… (good correlation) |
| 2300 PREDX | produces 6.8139 |
| 5.2 PREDY | produces 463.37 |
| NXT STAT PLOT SCATR | produce scattergram of y vs. x |
| STATL | show line for log fitting |



- To return to STAT menu use: NXT STAT
- To get your variable menu back use: VAR .

## Confidence intervals

Statistical inference is the process of making conclusions about a population based on  information from sample data.   In order for the sample data to be meaningful, the sample must be random, i.e., the selection of a particular sample must have the same probability as that of any other possible sample out of a given population.  The following are some terms relevant to the concept of random sampling:

- Population:  collection of all conceivable observations of a process or attribute of a component.
- Sample: sub-set of a population.
- Random sample: a sample representative of the population.
- Random variable: real-valued function defined on a sample space.  Could be discrete or continuous.

  If the population follows a certain probability distribution that depends on a parameter θ, a random sample of observations ($X_1$,$X_2$,$X_3$,... , $X_n$), of size n, can be used to estimate θ.

- Sampling distribution: the joint probability distribution of $X_1$,$X_2$,$X_3$,... , $X_n$.
- A statistic: any function of the observations that is quantifiable and does not contain any unknown parameters.  A statistic is a random variable that provides a means of estimation.
- Point estimation: when a single value of the parameter θ is provided.
- Confidence interval: a numerical interval that contains the parameter θ at a given level of probability.
- Estimator: rule or method of estimation of the parameter θ.
- Estimate: value that the estimator yields in a particular application.

Example 1 --  Let X represent the time (hours) required by a specific manufacturing process to be completed.  Given the following sample of values of  X:  2.2   2.5   2.1   2.3   2.2.   The population from where this sample is taken is the collection of all possible values of the process time, therefore, it is an infinite population.  Suppose that the population parameter we are trying to estimate is its mean value, $\mu$. We will use as an estimator the mean value of the sample, $\overline{X}$, defined by (a rule): $\overline{X} = \frac{1}{n} \cdot \sum_{i=1}^{n} X_i$.

For the sample under consideration, the estimate of $\mu$ is the sample statistic $\overline{x}$ = (2.2+2.5+2.1+2.3+2.2)/5 = 2.36. This single value of $\overline{X}$, namely $\overline{x}$ = 2.36, constitutes a point estimation of the population parameter $\mu$.

## Estimation of Confidence Intervals

The next level of inference from point estimation is interval estimation, i.e., instead of obtaining a single value of an estimator we provide two statistics, a and b, which define an interval containing the parameter $\theta$ with a certain level of probability. The end points of the interval are known as confidence limits, and the interval (a,b) is known as the confidence interval.

## Definitions

Let $(C_l,C_u)$ be a confidence interval containing an unknown parameter $\theta$.

- Confidence level or confidence coefficient is the quantity (1-$\alpha$), where 0 < $\alpha$ < 1, such that $P[C_l < \theta < C_u] = 1 - \alpha$, where P[ ] represents a probability (see Chapter 17). The previous expression defines the so-called two-sided confidence limits.
- A lower one-sided confidence interval is defined by $Pr[C_l < \theta] = 1 - \alpha$.
- An upper one-sided confidence interval is defined by $Pr[\theta < C_u] = 1 - \alpha$.
- The parameter $\alpha$ is known as the significance level. Typical values of $\alpha$ are 0.01, 0.05, 0.1, corresponding to confidence levels of 0.99, 0.95, and 0.90, respectively.

## Confidence intervals for the population mean when the population variance is known

Let $\overline{X}$ be the mean of a random sample of size n, drawn from an infinite population with known standard deviation $\sigma$. The 100(1-$\alpha$) % [i.e., 99%, 95%, 90%, etc.], central, two-sided confidence interval for the population mean $\mu$ is ( $\overline{X}-z_{\alpha/2}\cdot\sigma/\sqrt{n}$ , $\overline{X}+z_{\alpha/2}\cdot\sigma/\sqrt{n}$ ), where $z_{\alpha/2}$ is a standard normal variate that is exceeded with a probability of $\alpha$ /2. The standard error of the sample mean, $\overline{X}$, is $\cdot\sigma/\sqrt{n}$.

The one-sided upper and lower 100(1-$\alpha$) % confidence limits for the population mean $\mu$ are, respectively, $X+z_{\alpha}\cdot\sigma/\sqrt{n}$ , and $\bar{X}-z_{\alpha}\cdot\sigma/\sqrt{n}$ . Thus, a lower, one-sided, confidence interval is defined as $(-\infty\ , X+z_{\alpha}\cdot\sigma/\sqrt{n})$, and an upper, one-sided, confidence interval as $(X-z_{\alpha}\cdot\sigma/\sqrt{n},+\infty)$. Notice that in these last two intervals we use the value $z_{\alpha}$, rather than $z_{\alpha/2}$.

In general, the value $z_k$ in the standard normal distribution is defined as that value of z whose probability of exceedence is k, i.e., $\Pr[Z>z_k] = k$, or $\Pr[Z<z_k] = 1 - k$. The normal distribution was described in Chapter 17.

## Confidence intervals for the population mean when the population variance is unknown

Let $\bar{X}$ and S, respectively, be the mean and standard deviation of a random sample of size n, drawn from an infinite population that follows the normal distribution with unknown standard deviation $\sigma$. The $100\cdot(1-\alpha)$ % [i.e., 99%, 95%, 90%, etc.] central two-sided confidence interval for the population mean $\mu$, is $(\ \bar{X}- t_{n-1,\ \alpha/2}\cdot S/\sqrt{n}\ ,\ \ \bar{X}+ t_{n-1,\ \alpha/2}\cdot S/\sqrt{n}\ )$, where $t_{n-1,\ \alpha/2}$ is Student's t variate with $\nu = n-1$ degrees of freedom and probability $\alpha/2$ of exceedence.

The one-sided upper and lower $100\cdot$ (1-$\alpha$) % confidence limits for the population mean $\mu$ are, respectively,

$$X + t_{n-1,\ \alpha/2}\cdot S/\sqrt{n}\ ,\ \text{and}\ \ \bar{X}- t_{n-1,\ \alpha/2}\cdot S/\sqrt{n}.$$

### Small samples and large samples

The behavior of the Student's t distribution is such that for n>30, the distribution is indistinguishable from the standard normal distribution. Thus, for samples larger than 30 elements when the population variance is unknown, you can use the same confidence interval as when the population variance is known, but replacing $\sigma$ with S. Samples for which n>30 are typically referred to as large samples, otherwise they are small samples.

## Confidence interval for a proportion

A discrete random variable X follows a Bernoulli distribution if X can take only two values, X = 0 (failure), and X = 1 (success). Let X ~ Bernoulli(p), where p is the probability of success, then the mean value, or expectation, of X is E[X] = p, and its variance is Var[X] = p(1-p).

If an experiment involving X is repeated n times, and k successful outcomes are recorded, then an estimate of p is given by $p' = k/n$, while the standard error of p' is $\sigma_{p'} = \sqrt{}(p\cdot(1-p)/n)$ . In practice, the sample estimate for p, i.e., p' replaces p in the standard error formula.

For a large sample size, n>30, and $n\cdot p > 5$ and $n\cdot(1-p)>5$, the sampling distribution is very nearly normal. Therefore, the $100(1-\alpha)$ % central two-sided confidence interval for the population mean p is $(p'+z_{\alpha/2}\cdot\sigma_{p'},\ p'+z_{\alpha/2}\cdot\sigma_{p'})$. For a small sample (n<30), the interval can be estimated as $(p'-t_{n-1,\alpha/2}\cdot\sigma_{p'}, p'+t_{n-1,\alpha/2}\cdot\sigma_{p'})$.

## Sampling distribution of differences and sums of statistics

Let $S_1$ and $S_2$ be independent statistics from two populations based on samples of sizes $n_1$ and $n_2$, respectively. Also, let the respective means and standard errors of the sampling distributions of those statistics be $\mu_{S1}$ and $\mu_{S2}$, and $\sigma_{S1}$ and $\sigma_{S2}$, respectively. The differences between the statistics from the two populations, $S_1$-$S_2$, have a sampling distribution with mean $\mu_{S1-S2} = \mu_{S1} - \mu_{S2}$, and standard error $\sigma_{S1-S2} = (\sigma_{S1}^2 + \sigma_{S2}^2)^{1/2}$. Also, the sum of the statistics $T_1+T_2$ has a mean $\mu_{S1+S2} = \mu_{S1} + \mu_{S2}$, and standard error $\sigma_{S1+S2} = (\sigma_{S1}^2 + \sigma_{S2}^2)^{1/2}$.

Estimators for the mean and standard deviation of the difference and sum of the statistics $S_1$ and $S_2$ are given by:

$$\hat{\mu}_{S_1 \pm S_2} = \overline{X}_1 \pm \overline{X}_2, \qquad \hat{\sigma}_{S_1 \pm S_2} = \sqrt{\frac{\sigma_{S1}^2}{n_1} + \frac{\sigma_{S2}^2}{n_2}}$$

In these expressions, $\overline{X}_1$ and $\overline{X}_2$ are the values of the statistics $S_1$ and $S_2$ from samples taken from the two populations, and $\sigma_{S1}^2$ and $\sigma_{S2}^2$ are the variances of the populations of the statistics $S_1$ and $S_2$ from which the samples were taken.

## Confidence intervals for sums and differences of mean values

If the population variances $\sigma_1{}^2$ and $\sigma_2{}^2$ are known, the confidence intervals for the difference and sum of the mean values of the populations, i.e., $\mu_1 \pm \mu_2$, are given by:

$$\left( (\overline{X}_1 \pm X_2) - z_{\alpha/2} \cdot \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}, (\overline{X}_1 \pm X_2) + z_{\alpha/2} \cdot \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} \right)$$

For large samples, i.e., $n_1 > 30$ and $n_2 > 30$, and unknown, but equal, population variances $\sigma_1{}^2 = \sigma_2{}^2$, the confidence intervals for the difference and sum of the mean values of the populations, i.e., $\mu_1 \pm \mu_2$, are given by:

$$\left( (\overline{X}_1 \pm X_2) - z_{\alpha/2} \cdot \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}, (\overline{X}_1 \pm X_2) + z_{\alpha/2} \cdot \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}} \right).$$

If one of the samples is small, i.e., $n_1 < 30$ or $n_2 < 30$, and with unknown, but equal, population variances $\sigma_1{}^2 = \sigma_2{}^2$, we can obtain a "pooled" estimate of the variance of $\mu_1 \pm \mu_2$, as $s_p{}^2 = [(n_1-1)\cdot s_1{}^2+(n_2-1)\cdot s_2{}^2]/( n_1+n_2-2)$.

In this case, the centered confidence intervals for the sum and difference of the mean values of the populations, i.e., $\mu_1 \pm \mu_2$, are given by:

$$\left( (\overline{X}_1 \pm X_2) - t_{\nu,\alpha/2} \cdot s_p^2, (\overline{X}_1 \pm X_2) + t_{\nu,\alpha/2} \cdot s_p^2 \right)$$

where $\nu = n_1+n_2-2$ is the number of degrees of freedom in the Student's t distribution.

In the last two options we specify that the population variances, although unknown, must be equal. This will be the case in which the two samples are taken from the same population, or from two populations about which we suspect that they have the same population variance. However, if we have reason to believe that the two unknown population variances are different, we can use the following confidence interval

$$\left((\overline{X}_1 \pm X_2) - t_{\nu,\alpha/2} \cdot s^2_{\overline{X}_1 \pm \overline{X}_2},\ (\overline{X}_1 \pm X_2) + t_{\nu,\alpha/2} \cdot s^2_{\overline{X}_1 \pm \overline{X}_2}\right)$$

where the estimated standard deviation for the sum or difference is

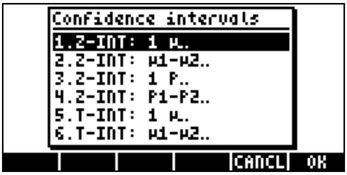
$$s_{\overline{X}_1 \pm \overline{X}_2} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

and n, the degrees of freedom of the t variate, are calculated using the integer value closest to

$$\nu = \frac{[(S_1^2/n_1) + (S_2^2/n_2)]^2}{[(S_1^2/n_1)/(n_1-1)] + [(S_2^2/n_2)/(n_2-1)]}$$

## Determining confidence intervals

The application **6. Conf Interval** can be accessed by using ↱ STAT ▲ OK. The application offers the following options:



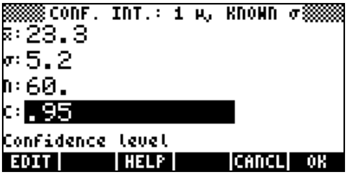
These options are to be interpreted as follows:

1. Z-INT: 1 μ.: Single sample confidence interval for the population mean, μ, with known population variance, or for large samples with unknown population variance.
2. Z-INT: μ1−μ2.: Confidence interval for the difference of the population means, $\mu_1$- $\mu_2$, with either known population variances, or for large samples with unknown population variances.

3. Z-INT: 1 p.: Single sample confidence interval for the proportion, p, for large samples with unknown population variance.
4. Z-INT: p1− p2.: Confidence interval for the difference of two proportions, $p_1$-$p_2$, for large samples with unknown population variances.
5. T-INT: 1 μ.: Single sample confidence interval for the population mean, μ, for small samples with unknown population variance.
6. T-INT: μ1−μ2.: Confidence interval for the difference of the population means, $\mu_1$- $\mu_2$, for small samples with unknown population variances.

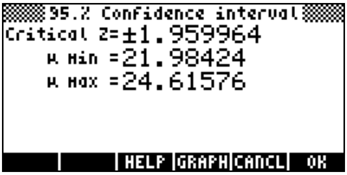
Example 1 – Determine the centered confidence interval for the mean of a population if a sample of 60 elements indicate that the mean value of the sample is $\bar{x}$ = 23.2, and its standard deviation is s = 5.2. Use α = 0.05. The confidence level is C = 1-α = 0.95.

Select case 1 from the menu shown above by pressing OK. Enter the values required in the input form as shown:


CONF. INT.: 1 μ, KNOWN σ
x̄:23.3
σ:5.2
n:60.
C:.95
Confidence level

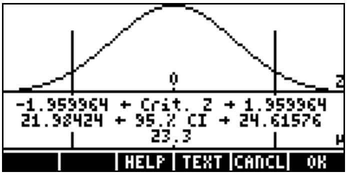
Press HELP to obtain a screen explaining the meaning of the confidence interval in terms of random numbers generated by a calculator. To scroll down the resulting screen use the down-arrow key ▼. Press OK when done with the help screen. This will return you to the screen shown above.

To calculate the confidence interval, press OK. The result shown in the calculator is:


95.% Confidence interval
Critical Z=±1.959964
μ Min =21.98424
μ Max =24.61576

The result indicates that a 95% confidence interval has been calculated. The Critical z value shown in the screen above corresponds to the values $\pm z_{\alpha/2}$ in the confidence interval formula ( $\bar{X}-z_{\alpha/2}\cdot\sigma/\sqrt{n}$ , $\bar{X}+z_{\alpha/2}\cdot\sigma/\sqrt{n}$ ). The values μ Min and μ Max are the lower and upper limits of this interval, i.e., μ Min = $\bar{X}-z_{\alpha/2}\cdot\sigma/\sqrt{n}$, and μ Max = $\bar{X}+z_{\alpha/2}\cdot\sigma/\sqrt{n}$.

Press GRAPH to see a graphical display of the confidence interval information:

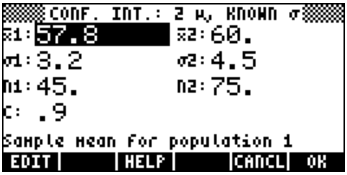


```
-1.959964 ← Crit. Z → 1.959964
21.98424 ← 95.% CI → 24.61576
23.3
```

The graph shows the standard normal distribution pdf (probability density function), the location of the critical points $\pm z_{\alpha/2}$, the mean value (23.2) and the corresponding interval limits (21.88424 and 24.51576). Press TEXT to return to the previous results screen, and/or press OK to exit the confidence interval environment. The results will be listed in the calculator's display.

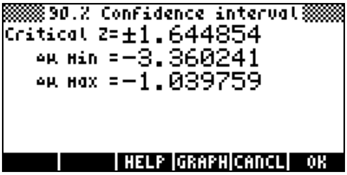
Example 2 -- Data from two samples (samples 1 and 2) indicate that $\bar{x}_1$ = 57.8 and $\bar{x}_2$ = 60.0. The sample sizes are $n_1$ = 45 and $n_2$ = 75. If it is known that the populations' standard deviations are $\sigma_1$ = 3.2, and $\sigma_2$ = 4.5, determine the 90% confidence interval for the difference of the population means, i.e., $\mu_1$- $\mu_2$.

Press ↱ STAT ▲ OK to access the confidence interval feature in the calculator. Press ▼ OK to select option 2. Z-INT: μ 1 – μ2.. Enter the following values:

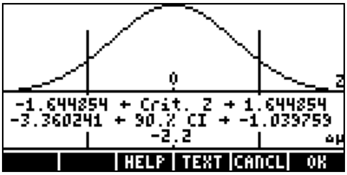


```
CONF. INT.: 2 μ, KNOWN σ
x̄1: 57.8          x̄2: 60.
σ1: 3.2           σ2: 4.5
n1: 45.           n2: 75.
C: .9
Sample mean for population 1
```

When done, press OK. The results, as text and graph, are shown below:



```
90.% Confidence interval
Critical Z=±1.644854
   Δμ Min =-3.360241
   Δμ Max =-1.039759
```

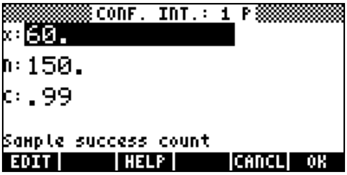


0
Z
-1.644854 ← Crit. Z → 1.644854
-3.360241 ← 90.% CI → -1.039759
-2.2
Δμ
HELP
TEXT
CANCL
OK

The variable Δμ represents μ 1 – μ2.

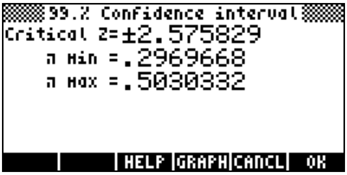
Example 3 – A survey of public opinion indicates that in a sample of 150 people 60 favor increasing property taxes to finance some public projects. Determine the 99% confidence interval for the population proportion that would favor increasing taxes.

Press ↱ _STAT_ ▲ OK to access the confidence interval feature in the calculator. Press ▼ ▼ OK to select option 3. Z-INT: μ 1 – μ2.. Enter the following values:

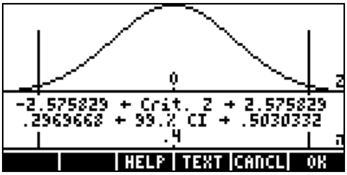


```
CONF. INT.: 1 P
x: 60.
n: 150.
C: .99
Sample success count
```

When done, press OK. The results, as text and graph, are shown below:

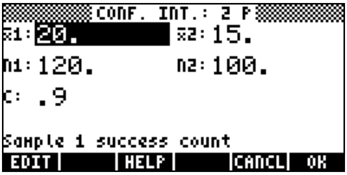


```
99.% Confidence interval
Critical Z=±2.575829
    π Min =.2969668
    π Max =.5030332
```



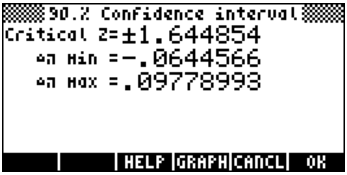
Example 4 -- Determine a 90% confidence interval for the difference between two proportions if sample 1 shows 20 successes out of 120 trials, and sample 2 shows 15 successes out of 100 trials.

Press ↱ STAT ▲ OK to access the confidence interval feature in the calculator. Press ▼▼▼ OK to select option 4. Z-INT: p1 – p2.. Enter the following values:

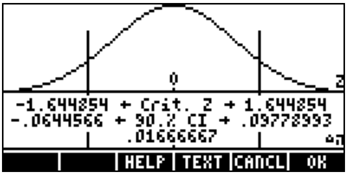


```
CONF. INT.: 2 P
x1: 20.        x2: 15.
n1: 120.       n2: 100.
C: .9

Sample 1 success count
```

When done, press OK. The results, as text and graph, are shown below:



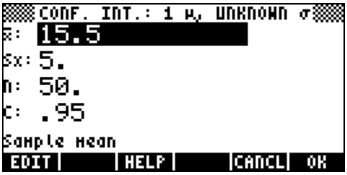
```
90.% Confidence interval
Critical Z=±1.644854
   Δπ Min =-.0644566
   Δπ Max =.09778993
```



```
-1.644854 ← Crit. Z → 1.644854
-.0644566 ← 90.% CI → .09778993
           .01666667
```

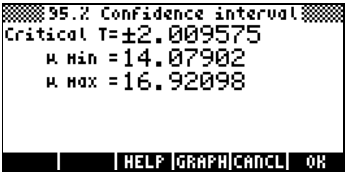
Example 5 – Determine a 95% confidence interval for the mean of the population if a sample of 50 elements has a mean of 15.5 and a standard deviation of 5. The population's standard deviation is unknown.

Press ↱ STAT ▲ OK to access the confidence interval feature in the calculator. Press ▲ ▲ OK to select option 5. T-INT: μ. Enter the following values:

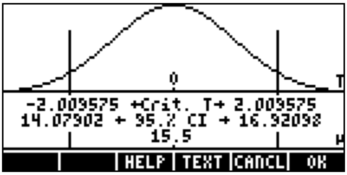


```
CONF. INT.: 1 μ, UNKNOWN σ
x̄: 15.5
Sx: 5.
n: 50.
C: .95
Sample mean
```

When done, press OK. The results, as text and graph, are shown below:



```
95.% Confidence interval
Critical T=±2.009575
    μ Min =14.07902
    μ Max =16.92098
```

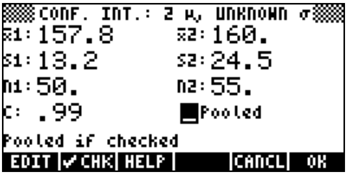


```
-2.009575 ←Crit. T→ 2.009575
14.07902 ← 95.% CI → 16.92098
15.5
```

The figure shows the Student's t pdf for ν = 50 – 1 = 49 degrees of freedom.

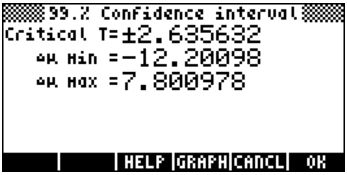
Example 6 -- Determine the 99% confidence interval for the difference in means of two populations given the sample data: $\bar{x}_1$ = 157.8 , $\bar{x}_2$ = 160.0, $n_1$ = 50, $n_2$ = 55. The populations standard deviations are $s_1$ = 13.2, $s_2$ = 24.5.

Press ↱ STAT ▲ OK to access the confidence interval feature in the calculator. Press ▲ OK to select option 6. T-INT: μ1–μ2.. Enter the following values:

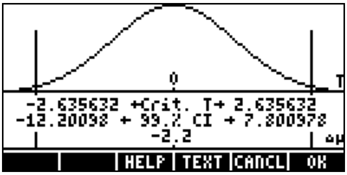


```
CONF. INT.: 2 μ, UNKNOWN σ
x̄1:157.8       x̄2:160.
s1:13.2        s2:24.5
n1:50.         n2:55.
C: .99         _Pooled
Pooled if checked
```

hen done, press OK. The results, as text and graph, are shown below:

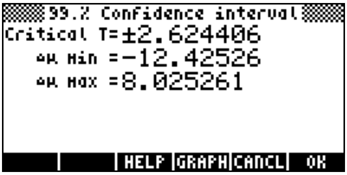


```
99.% Confidence interval
Critical T=±2.635632
   Δμ Min =-12.20098
   Δμ Max =7.800978
```

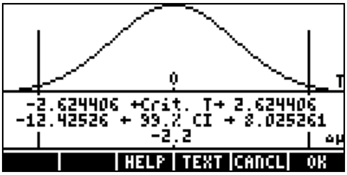


```
-2.635632 ←Crit. T→ 2.635632
-12.20098 ← 99.% CI → 7.800978
-2.2
```

These results assume that the values $s_1$ and $s_2$ are the population standard deviations. If these values actually represent the samples' standard deviations, you should enter the same values as before, but with the option `_pooled` selected. The results now become:



```
99.% Confidence interval
Critical T=±2.624406
   Δμ Min =-12.42526
   Δμ Max =8.025261
```



## Confidence intervals for the variance

To develop a formula for the confidence interval for the variance, first we introduce the sampling distribution of the variance: Consider a random sample $X_1$, $X_2$ ..., $X_n$ of independent normally-distributed variables with mean μ, variance $\sigma^2$, and sample mean $\overline{X}$. The statistic

$$\hat{S}^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (X_i - \overline{X})^2,$$

is an unbiased estimator of the variance $\sigma^2$.

The quantity $(n-1) \cdot \dfrac{\hat{S}^2}{\sigma^2} = \sum_{i=1}^{n} (X_i - \overline{X})^2$, has a $\chi_{n-1}{}^2$ (chi-square) distribution with ν = n-1 degrees of freedom. The (1-α)·100 % two-sided confidence interval is found from

$$\Pr[\chi^2_{n-1,1-\alpha/2} < (n-1)\cdot S^2/\sigma^2 < \chi^2_{n-1,\alpha/2}] = 1- \alpha.$$

The confidence interval for the population variance $\sigma^2$ is therefore,

$$[(n-1)\cdot S^2/\ \chi^2_{n-1,\alpha/2}\,;\ (n-1)\cdot S^2/\ \chi^2_{n-1,1-\alpha/2}].$$

where $\chi^2_{n-1,\alpha/2}$ , and $\chi^2_{n-1,1-\alpha/2}$ are the values that a $\chi^2$ variable, with ν = n-1 degrees of freedom, exceeds with probabilities α/2 and 1- α /2, respectively.

The one-sided upper confidence limit for $\sigma^2$ is defined as $(n-1)\cdot S^2/\ \chi^2_{n-1,1-\alpha}$.

Example 1 – Determine the 95% confidence interval for the population variance $\sigma^2$ based on the results from a sample of size n = 25 that indicates that the sample variance is $s^2$ = 12.5.

In Chapter 17 we use the numerical solver to solve the equation $\alpha$ = UTPC($\gamma$,x). In this program, $\gamma$ represents the degrees of freedom (n-1), and $\alpha$ represents the probability of exceeding a certain value of x ($\chi^2$), i.e., Pr[$\chi^2 > \chi_\alpha^2$] = $\alpha$.

For the present example, $\alpha$ = 0.05, $\gamma$ = 24 and $\alpha$ = 0.025. Solving the equation presented above results in $\chi^2_{n-1,\alpha/2} = \chi^2_{24,0.025}$ = 39.3640770266.

On the other hand, the value $\chi^2_{n-1,\alpha/2} = \chi^2_{24,0.975}$ is calculated by using the values $\gamma$ = 24 and $\alpha$ = 0.975. The result is $\chi^2_{n-1,1-\alpha/2} = \chi^2_{24,0.975}$ = 12.4011502175.

The lower and upper limits of the interval will be (Use ALG mode for these calculations):

$$(n\text{-}1)\cdot S^2/\ \chi^2_{n-1,\alpha/2} = (25\text{-}1)\cdot 12.5/39.3640770266 = 7.62116179676$$

$$(n\text{-}1)\cdot S^2/\ \chi^2_{n-1,1-\alpha/2} = (25\text{-}1)\cdot 12.5/12.4011502175 = 24.1913044144$$

Thus, the 95% confidence interval for this example is:

$$7.62116179676 < \sigma^2 < 24.1913044144.$$

## Hypothesis testing

A hypothesis is a declaration made about a population (for instance, with respect to its mean). Acceptance of the hypothesis is based on a statistical test on a sample taken from the population. The consequent action and decision-making are called hypothesis testing.

The process of hypothesis testing consists on taking a random sample from the population and making a statistical hypothesis about the population. If the observations do not support the model or theory postulated, the hypothesis is rejected. However, if the observations are in agreement, then hypothesis is not rejected, but it is not necessarily accepted. Associated with the decision is a level of significance $\alpha$.

## Procedure for testing hypotheses

The procedure for hypothesis testing involves the following six steps:

1. Declare a null hypothesis, $H_0$. This is the hypothesis to be tested. For example, $H_0$: $\mu_1$-$\mu_2$ $\neq$ 0, i.e., we hypothesize that the mean value of population 1 and the mean value of population 2 are the same. If $H_0$ is true, any observed difference in means is attributed to errors in random sampling.
2. Declare an alternate hypothesis, $H_1$. For the example under consideration, it could be $H_1$: $\mu_1$-$\mu_2$ $\neq$ 0 [Note: this is what we really want to test.]
3. Determine or specify a test statistic, T. In the example under consideration, T will be based on the difference of observed means, $\bar{X}_1$- $\bar{X}_2$.
4. Use the known (or assumed) distribution of the test statistic, T.
5. Define a rejection region (the critical region, R) for the test statistic based on a pre-assigned significance level $\alpha$.
6. Use observed data to determine whether the computed value of the test statistic is within or outside the critical region. If the test statistic is within the critical region, then we say that the quantity we are testing is significant at the 100$\alpha$ percent level.

**Notes:**

1. For the example under consideration, the alternate hypothesis $H_1$: $\mu_1$-$\mu_2$ $\neq$ 0 produces what is called a two-tailed test. If the alternate hypothesis is $H_1$: $\mu_1$-$\mu_2$ > 0 or $H_1$: $\mu_1$-$\mu_2$ < 0, then we have a one-tailed test.

2. The probability of rejecting the null hypothesis is equal to the level of significance, i.e., $\Pr[T\in R|H_0]=\alpha$. The notation $\Pr[A|B]$ represents the conditional probability of event A given that event B occurs.

## Errors in hypothesis testing

In hypothesis testing we use the terms errors of Type I and Type II to define the cases in which a true hypothesis is rejected or a false hypothesis is accepted (not rejected), respectively. Let T = value of test statistic, R = rejection region, A = acceptance region, thus, $R\cap A = \varnothing$, and $R\cup A = \Omega$, where $\Omega$ = the parameter space for T, and $\varnothing$ = the empty set. The probabilities of making an error of Type I or of Type II are as follows:

Rejecting a true hypothesis, Pr[Type I error] = $\Pr[T \in R | H_0] = \alpha$
Not rejecting a false hypothesis, Pr[Type II error] = $\Pr[T \in A | H_1] = \beta$

Now, let's consider the cases in which we make the correct decision:

Not rejecting a true hypothesis, Pr[Not(Type I error)] = $\Pr[T \in A | H_0] = 1 - \alpha$

Rejecting a false hypothesis, Pr[Not(Type II error)] = $\Pr [T \in R | H_1] = 1 - \beta$

The complement of $\beta$ is called the power of the test of the null hypothesis $H_0$ vs. the alternative $H_1$. The power of a test is used, for example, to determine a minimum sample size to restrict errors.

**Selecting values of $\alpha$ and $\beta$**

A typical value of the level of significance (or probability of Type I error) is $\alpha = 0.05$, (i.e., incorrect rejection once in 20 times on the average). If the consequences of a Type I error are more serious, choose smaller values of $\alpha$, say 0.01 or even 0.001.

The value of $\beta$, i.e., the probability of making an error of Type II, depends on $\alpha$, the sample size n, and on the true value of the parameter tested. Thus, the value of $\beta$ is determined after the hypothesis testing is performed. It is customary to draw graphs showing $\beta$, or the power of the test ($1-\beta$), as a function of the true value of the parameter tested. These graphs are called operating characteristic curves or power function curves, respectively.

## Inferences concerning one mean

Two-sided hypothesis

The problem consists in testing the null hypothesis $H_o$: $\mu = \mu_o$, against the alternative hypothesis, $H_1$: $\mu \neq \mu_o$ at a level of confidence $(1-\alpha)100\%$, or significance level $\alpha$, using a sample of size n with a mean $\bar{x}$ and a standard deviation s. This test is referred to as a two-sided or two-tailed test. The procedure for the test is as follows:

First, we calculate the appropriate statistic for the test ($t_o$ or $z_o$) as follows:

- If n < 30 and the standard deviation of the population, σ, is known, use the z-statistic: $z_o = \frac{\bar{x} - \mu_o}{\sigma / \sqrt{n}}$
- If n > 30, and σ is known, use $z_o$ as above. If σ is not known, replace s for σ in $z_o$, i.e., use $z_o = \frac{\bar{x} - \mu_o}{s / \sqrt{n}}$
- If n < 30, and σ is unknown, use the t-statistic $t_o = \frac{\bar{x} - \mu_o}{s / \sqrt{n}}$, with ν = n - 1 degrees of freedom.

Then, calculate the P-value (a probability) associated with either $z_o$ or $t_o$ , and compare it to α to decide whether or not to reject the null hypothesis. The P-value for a two-sided test is defined as either

$$\text{P-value} = P(|z|>|z_o|), \text{ or, } \text{P-value} = P(|t|>|t_o|).$$

The criteria to use for hypothesis testing is:

- Reject $H_o$ if P-value < α
- Do not reject $H_o$ if P-value > α.

The P-value for a two-sided test can be calculated using the probability functions in the calculator as follows:

- If using z, P-value = $2\cdot$UTPN(0,1,$|z_o|$)
- If using t, P-value = $2\cdot$UTPT(ν,$|t_o|$)

Example 1 -- Test the null hypothesis $H_o$: μ = 22.5 ( = $\mu_o$), against the alternative hypothesis, $H_1$: μ ≠22.5, at a level of confidence of 95% i.e., α = 0.05, using a sample of size n = 25 with a mean $\bar{x}$ = 22.0 and a standard deviation s = 3.5. We assume that we don't know the value of the population standard deviation, therefore, we calculate a t statistic as follows:

$$t_o = \frac{\bar{x} - \mu_o}{s/\sqrt{n}} = \frac{22.0 - 22.5}{3.5/\sqrt{25}} = -0.7142$$

The corresponding P-value, for n = 25 - 1 = 24 degrees of freedom is

P-value = 2·UTPT(24,-0.7142) = 2·0.7590 = 1.5169,

since 1.5169 > 0.05, i.e., P-value > $\alpha$, we cannot reject the null hypothesis $H_o$: $\mu$ = 22.0.

One-sided hypothesis

The problem consists in testing the null hypothesis $H_o$: $\mu = \mu_o$, against the alternative hypothesis, $H_1$: $\mu > \mu_o$ or $H_1$: $\mu < \mu_o$ at a level of confidence (1-$\alpha$)100%, or significance level $\alpha$, using a sample of size n with a mean $\bar{x}$ and a standard deviation s. This test is referred to as a one-sided or one-tailed test. The procedure for performing a one-side test starts as in the two-tailed test by calculating the appropriate statistic for the test ($t_o$ or $z_o$) as indicated above.

Next, we use the P-value associated with either $z_o$ or $t_o$ , and compare it to $\alpha$ to decide whether or not to reject the null hypothesis. The P-value for a two-sided test is defined as either

P-value = P(z > $|z_o|$), or, P-value = P(t > $|t_o|$).

The criteria to use for hypothesis testing is:

- Reject $H_o$ if P-value < $\alpha$
- Do not reject $H_o$ if P-value > $\alpha$.

Notice that the criteria are exactly the same as in the two-sided test. The main difference is the way that the P-value is calculated. The P-value for a one-sided test can be calculated using the probability functions in the calculator as follows:

- If using z, P-value = UTPN(0,1,$z_o$)
- If using t, P-value = UTPT($\nu$,$t_o$)

Example 2 -- Test the null hypothesis $H_o$: $\mu$ = 22.0 ( = $\mu_o$), against the alternative hypothesis, $H_1$: $\mu$ >22.5 at a level of confidence of 95% i.e., $\alpha$ = 0.05, using a sample of size n = 25 with a mean $\bar{x}$ = 22.0 and a standard deviation s = 3.5. Again, we assume that we don't know the value of the population standard deviation, therefore, the value of the t statistic is the same as in the two-sided test case shown above, i.e., $t_o$ = -0.7142, and P-value, for $\nu$ = 25 - 1 = 24 degrees of freedom is

P-value = UTPT(24, |-0.7142|) = UTPT(24,0.7124) = 0.2409,

since 0.2409 > 0.05, i.e., P-value > $\alpha$, we cannot reject the null hypothesis $H_o$: $\mu$ = 22.0.

## Inferences concerning two means

The null hypothesis to be tested is $H_o$: $\mu_1$-$\mu_2$ = $\delta$, at a level of confidence (1-$\alpha$)100%, or significance level $\alpha$, using two samples of sizes, $n_1$ and $n_2$, mean values $\bar{x}_1$ and $\bar{x}_2$, and standard deviations $s_1$ and $s_2$. If the populations standard deviations corresponding to the samples, $\sigma_1$ and $\sigma_2$, are known, or if $n_1$ > 30 and $n_2$ > 30 (large samples), the test statistic to be used is

$$z_o = \frac{(\bar{x}_1 - \bar{x}_2) - \delta}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

If $n_1$ < 30 or $n_2$ < 30 (at least one small sample), use the following test statistic:

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - \delta}{\sqrt{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}} \sqrt{\frac{n_1 n_2 (n_1 + n_2 - 2)}{n_1 + n_2}}$$

Two-sided hypothesis

If the alternative hypothesis is a two-sided hypothesis, i.e., $H_1$: $\mu_1$-$\mu_2 \neq \delta$, The P-value for this test is calculated as

- If using z, P-value = 2·UTPN(0,1, $|z_o|$)
- If using t, P-value = 2·UTPT($\nu$,$|t_o|$)

with the degrees of freedom for the t-distribution given by $\nu = n_1 + n_2 - 2$. The test criteria are

- Reject $H_o$ if P-value < $\alpha$
- Do not reject $H_o$ if P-value > $\alpha$.

One-sided hypothesis

If the alternative hypothesis is a two-sided hypothesis, i.e., $H_1$: $\mu_1$-$\mu_2 < \delta$, or, $H_1$: $\mu_1$-$\mu_2 < \delta$,, the P-value for this test is calculated as:

- If using z, P-value = UTPN(0,1, $|z_o|$)
- If using t, P-value = UTPT($\nu$,$|t_o|$)

The criteria to use for hypothesis testing is:

- Reject $H_o$ if P-value < $\alpha$
- Do not reject $H_o$ if P-value > $\alpha$.

## Paired sample tests

When we deal with two samples of size n with paired data points, instead of testing the null hypothesis, $H_o$: $\mu_1$-$\mu_2 = \delta$, using the mean values and standard deviations of the two samples, we need to treat the problem as a single sample of the differences of the paired values. In other words, generate a new random variable $X = X_1$-$X_2$, and test $H_o$: $\mu = \delta$, where $\mu$ represents the mean of the population for X. Therefore, you will need to obtain $\bar{x}$ and s for the sample of values of x. The test should then proceed as a one-sample test using the methods described earlier.

## Inferences concerning one proportion

Suppose that we want to test the null hypothesis, $H_0$: p = $p_0$, where p represents the probability of obtaining a successful outcome in any given repetition of a Bernoulli trial. To test the hypothesis, we perform n repetitions of the experiment, and find that k successful outcomes are recorded. Thus, an estimate of p is given by p' = k/n.

The variance for the sample will be estimated as $s_p^2 = p'(1-p')/n = k\cdot(n-k)/n^3$.

Assume that the Z score, $Z = (p-p_0)/s_p$, follows the standard normal distribution, i.e., Z ~ N(0,1). The particular value of the statistic to test is $z_0 = (p'-p_0)/s_p$.

Instead of using the P-value as a criterion to accept or not accept the hypothesis, we will use the comparison between the critical value of z0 and the value of z corresponding to α or α/2.

Two-tailed test

If using a two-tailed test we will find the value of $z_{\alpha/2}$, from

$$\Pr[Z> z_{\alpha/2}] = 1-\Phi(z_{\alpha/2}) = \alpha/2, \text{ or } \Phi(z_{\alpha/2}) = 1- \alpha/2,$$

where Φ(z) is the cumulative distribution function (CDF) of the standard normal distribution (see Chapter 17).

Reject the null hypothesis, $H_0$, if $z_0 > z_{\alpha/2}$, or if $z_0 < - z_{\alpha/2}$.

In other words, the rejection region is $R = \{ |z_0| > z_{\alpha/2} \}$, while the acceptance region is $A = \{|z_0| < z_{\alpha/2} \}$.

One-tailed test

If using a one-tailed test we will find the value of S, from

$$\Pr[Z> z_{\alpha}] = 1-\Phi(z_{\alpha}) = \alpha, \text{ or } \Phi(z_{\alpha}) = 1- \alpha,$$

Reject the null hypothesis, $H_0$, if $z_0 > z_\alpha$, and $H_1$: $p>p_0$, or if $z_0 < - z_\alpha$, and $H_1$: $p<p_0$.

## Testing the difference between two proportions

Suppose that we want to test the null hypothesis, $H_0$: $p_1$-$p_2$ = $p_0$, where the p's represents the probability of obtaining a successful outcome in any given repetition of a Bernoulli trial for two populations 1 and 2. To test the hypothesis, we perform $n_1$ repetitions of the experiment from population 1, and find that $k_1$ successful outcomes are recorded. Also, we find $k_2$ successful outcomes out of $n_2$ trials in sample 2. Thus, estimates of $p_1$ and $p_2$ are given, respectively, by $p_1' = k_1/n_1$, and $p_2' = k_2/n_2$.

The variances for the samples will be estimated, respectively, as

$$s_1^2 = p_1'(1\text{-}p_1')/n_1 = k_1\cdot(n_1\text{-}k_1)/n_1^3, \text{ and } s_2^2 = p_2'(1\text{-}p_2')/n_2 = k_2\cdot(n_2\text{-}k_2)/n_2^3.$$

And the variance of the difference of proportions is estimated from: $s_p^2 = s_1^2 + s_2^2$ .

Assume that the Z score, $Z = (p_1\text{-}p_2\text{-}p_0)/s_p$, follows the standard normal distribution, i.e., $Z \sim N(0,1)$. The particular value of the statistic to test is $z_0 = (p_1'\text{-}p_2'\text{-}p_0)/s_p$.

Two-tailed test

If using a two-tailed test we will find the value of $z_{\alpha/2}$, from

$$\Pr[Z> z_{\alpha/2}] = 1\text{-}\Phi(z_{\alpha/2}) = \alpha/2, \text{ or } \Phi(z_{\alpha/2}) = 1\text{-} \alpha/2,$$

where $\Phi(z)$ is the cumulative distribution function (CDF) of the standard normal distribution.

Reject the null hypothesis, $H_0$, if $z_0 > z_{\alpha/2}$, or if $z_0 < - z_{\alpha/2}$.

In other words, the rejection region is $R = \{ |z_0| > z_{\alpha/2} \}$, while the acceptance region is $A = \{|z_0| < z_{\alpha/2} \}$.

One-tailed test

If using a one-tailed test we will find the value of $z_a$, from

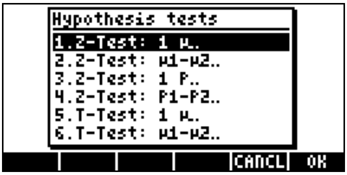
$$\Pr[Z> z_\alpha] = 1-\Phi(z_\alpha) = \alpha, \text{ or } \Phi(z_\alpha) = 1- \alpha,$$

Reject the null hypothesis, $H_0$, if $z_0 > z_\alpha$, and $H_1$: $p_1-p_2 > p_0$, or if $z_0 < - z_\alpha$, and $H_1$: $p_1-p_2 < p_0$.

## Hypothesis testing using pre-programmed features

The calculator provides with hypothesis testing procedures under application *5. Hypoth. tests..* can be accessed by using ↱ STAT ▲ ▲ OK.

As with the calculation of confidence intervals, discussed earlier, this program offers the following 6 options:



These options are interpreted as in the confidence interval applications:

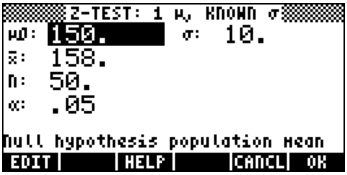
1. Z-Test: 1 μ.: Single sample hypothesis testing for the population mean, μ, with known population variance, or for large samples with unknown population variance.
2. Z-Test: μ1–μ2.: Hypothesis testing for the difference of the population means, $\mu_1$- $\mu_2$, with either known population variances, or for large samples with unknown population variances.
3. Z-Test: 1 p.: Single sample hypothesis testing for the proportion, p, for large samples with unknown population variance.
4. Z-Test: p1– p2.: Hypothesis testing for the difference of two proportions, $p_1$-$p_2$, for large samples with unknown population variances.
5. T-Test: 1 μ.: Single sample hypothesis testing for the population mean, μ, for small samples with unknown population variance.
6. T-Test: μ1–μ2.: Hypothesis testing for the difference of the population means, $\mu_1$- $\mu_2$, for small samples with unknown population variances.

Try the following exercises:

Example 1 – For $\mu_0$ = 150, $\sigma$ = 10, $\bar{x}$ = 158, n = 50, for $\alpha$ = 0.05, test the hypothesis $H_0$: $\mu = \mu_0$, against the alternative hypothesis, $H_1$: $\mu \neq \mu_0$.

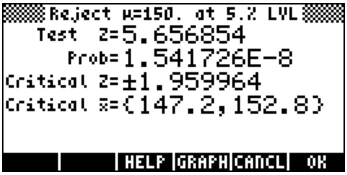
Press ↱ STAT ▲ ▲ OK to access the hypothesis testing feature in the calculator. Press OK to select option 1. Z-Test: 1 μ.

Enter the following data and press OK:



```
Z-TEST: 1 μ, KNOWN σ
μ0: 150.        σ: 10.
x̄: 158.
n: 50.
α: .05
Null hypothesis population mean
EDIT |     | HELP |     | CANCL | OK
```

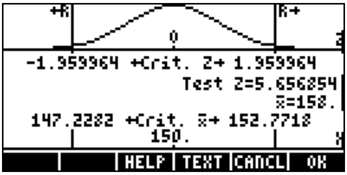
You are then asked to select the alternative hypothesis. Select μ ≠150, and press OK. The result is:



```
Reject μ=150. at 5.% LVL
  Test  Z=5.656854
     Prob=1.541726E-8
Critical Z=±1.959964
Critical x̄={147.2,152.8}

     |     | HELP |GRAPH|CANCL| OK
```

Then, we reject $H_0$: $\mu$ = 150, against $H_1$: $\mu \neq$ 150. The test z value is $z_0$ = 5.656854. The P-value is 1.54×$10^{-8}$. The critical values of $\pm z_{\alpha/2}$ = ±1.959964, corresponding to critical $\bar{x}$ range of {147.2 152.8}.

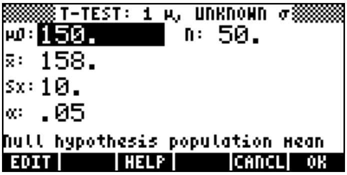
This information can be observed graphically by pressing the soft-menu key GRAPH:



```
-1.959964 ←Crit. Z→ 1.959964
              Test Z=5.656854
                       x̄=158.
 147.2282 ←Crit. x̄→ 152.7718
           150.
     |     | HELP | TEXT |CANCL| OK
```

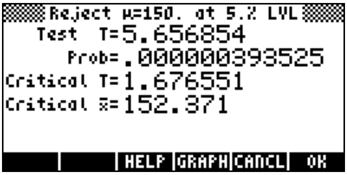
Example 2 -- For $\mu_0$ = 150, $\bar{x}$ = 158, s = 10, n = 50, for $\alpha$ = 0.05, test the hypothesis $H_0$: $\mu$ = $\mu_0$, against the alternative hypothesis, $H_1$: $\mu$ > $\mu_0$. The population standard deviation, $\sigma$, is not known.

Press ↱ STAT ▲ ▲ OK to access the hypothesis testing feature in the calculator. Press ▲ ▲ OK to select option 5. T-Test: 1 $\mu$.:
Enter the following data and press OK:



```
T-TEST: 1 μ, UNKNOWN σ
μ0: 150.        n: 50.
x̄: 158.
Sx: 10.
α: .05
null hypothesis population mean
EDIT |     | HELP |     |CANCL| OK
```

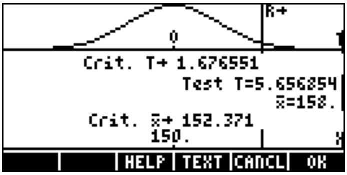
Select the alternative hypothesis, $H_1$: $\mu$ > 150, and press OK. The result is:



```
Reject μ=150. at 5.% LVL
   Test  T=5.656854
      Prob=.000000393525
Critical T=1.676551
Critical x̄=152.371

     |     | HELP |GRAPH|CANCL| OK
```

We reject the null hypothesis, $H_0$: $\mu_0$ = 150, against the alternative hypothesis, $H_1$: $\mu$ > 150. The test t value is $t_0$ = 5.656854, with a P-value = 0.000000393525. The critical value of t is $t_\alpha$ = 1.676551, corresponding to a critical $\bar{x}$ = 152.371.

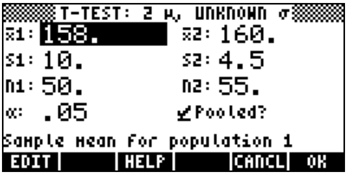
Press GRAPH to see the results graphically as follows:



```
Crit. T+ 1.676551
          Test T=5.656854
                  x̄=158.
Crit. x̄+ 152.371
      150.
     |     | HELP | TEXT |CANCL| OK
```

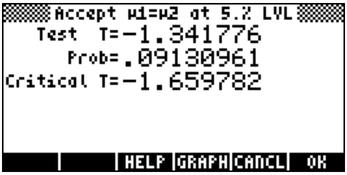
Example 3 – Data from two samples show that $\bar{x}_1$ = 158, $\bar{x}_1$ = 160, $s_1$ = 10, $s_2$ = 4.5, n1 = 50, and $n_2$ = 55. For $\alpha$ = 0.05, and a "pooled" variance, test the hypothesis $H_0$: $\mu_1-\mu_2 = 0$, against the alternative hypothesis, $H_1$: $\mu_1-\mu_2 < 0$.

Press ↱ STAT ▲ ▲ OK to access the hypothesis testing feature in the calculator. Press ▲ OK to select option 6. T-Test: μ1−μ2.: Enter the following data and press OK:



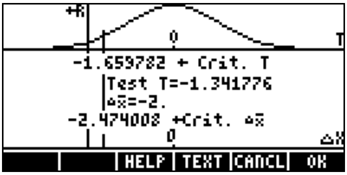
```
T-TEST: 2 μ, UNKNOWN σ
x̄1: 158.        x̄2: 160.
s1: 10.         s2: 4.5
n1: 50.         n2: 55.
α:  .05         ✓Pooled?
Sample mean for population 1
```

Select the alternative hypothesis μ1< μ2, and press OK. The result is



```
Accept μ1=μ2 at 5.% LVL
   Test  T=-1.341776
      Prob=.09130961
Critical T=-1.659782
```

Thus, we accept (more accurately, we do not reject) the hypothesis: $H_0$: $\mu_1-\mu_2 = 0$, or $H_0$: $\mu_1=\mu_2$, against the alternative hypothesis $H_1$: $\mu_1-\mu_2 < 0$, or $H_1$: $\mu_1=\mu_2$. The test t value is $t_0$ = -1.341776, with a P-value = 0.09130961, and critical t is $-t_\alpha$ = -.1659782. The graphical results are:



```
-1.659782 ← Crit. T
   Test T=-1.341776
   Δx̄=-2.
-2.474008 ←Crit. Δx̄
```

These three examples should be enough to understand the operation of the hypothesis testing pre-programmed feature in the calculator.

## Inferences concerning one variance

The null hypothesis to be tested is , $H_o$: $\sigma^2 = \sigma_o^2$, at a level of confidence (1-$\alpha$)100%, or significance level $\alpha$, using a sample of size n, and variance $s^2$. The test statistic to be used is a chi-squared test statistic defined as

$$\chi_o^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

Depending on the alternative hypothesis chosen, the P-value is calculated as follows:

- $H_1$: $\sigma^2 < \sigma_o^2$, P-value = $P(\chi^2<\chi_o^2)$ = 1-UTPC($\nu,\chi_o^2$)
- $H_1$: $\sigma^2 > \sigma_o^2$, P-value = $P(\chi^2>\chi_o^2)$ = UTPC($\nu,\chi_o^2$)
- $H_1$: $\sigma^2 \neq \sigma_o^2$, P-value =2·min[$P(\chi^2<\chi_o^2)$, $P(\chi^2>\chi_o^2)$] = 2·min[1-UTPC($\nu,\chi_o^2$), UTPC($\nu,\chi_o^2$)]

where the function min[x,y] produces the minimum value of x or y (similarly, max[x,y] produces the maximum value of x or y). UTPC($\nu$,x) represents the calculator's upper-tail probabilities for $\nu$ = n - 1 degrees of freedom.

The test criteria are the same as in hypothesis testing of means, namely,

- Reject $H_o$ if P-value < $\alpha$
- Do not reject $H_o$ if P-value > $\alpha$.

Please notice that this procedure is valid only if the population from which the sample was taken is a Normal population.

Example 1 -- Consider the case in which $\sigma_o^2$ = 25, $\alpha$=0.05, n = 25, and $s^2$ = 20, and the sample was drawn from a normal population. To test the hypothesis, $H_o$: $\sigma^2 = \sigma_o^2$, against $H_1$: $\sigma^2 < \sigma_o^2$, we first calculate

$$\chi_o^2 = \frac{(n-1)s^2}{\sigma_0^2} = \frac{(25-1)\cdot 20}{25} = 189.2$$

With $\nu$ = n - 1 = 25 - 1 = 24 degrees of freedom, we calculate the P-value as,

P-value = $P(\chi^2<19.2)$ = 1-UTPC(24,19.2) = 0.2587…

Since, 0.2587… > 0.05, i.e., P-value > $\alpha$, we cannot reject the null hypothesis, $H_o$: $\sigma^2$ =25(= $\sigma_o^2$).

## Inferences concerning two variances

The null hypothesis to be tested is , $H_o$: $\sigma_1^2 = \sigma_2^2$, at a level of confidence (1-$\alpha$)100%, or significance level $\alpha$, using two samples of sizes, $n_1$ and $n_2$, and variances $s_1^2$ and $s_2^2$. The test statistic to be used is an F test statistic defined as

$$F_o = \frac{s_N^2}{s_D^2}$$

where $s_N^2$ and $s_D^2$ represent the numerator and denominator of the F statistic, respectively. Selection of the numerator and denominator depends on the alternative hypothesis being tested, as shown below. The corresponding F distribution has degrees of freedom, $\nu_N = n_N$-1, and $\nu_D = n_D$-1, where $n_N$ and $n_D$, are the sample sizes corresponding to the variances $s_N^2$ and $s_D^2$, respectively.

The following table shows how to select the numerator and denominator for $F_o$ depending on the alternative hypothesis chosen:

| *Alternative hypothesis* | *Test statistic* | *Degrees of freedom* |
|---|---|---|
| $H_1$: $\sigma_1^2 < \sigma_2^2$ (one-sided) | $F_o = s_2^2/s_1^2$ | $\nu_N = n_2$-1, $\nu_D = n_1$-1 |
| $H_1$: $\sigma_1^2 > \sigma_2^2$ (one-sided) | $F_o = s_1^2/s_2^2$ | $\nu_N = n_1$-1, $\nu_D = n_2$-1 |
| $H_1$: $\sigma_1^2 \neq \sigma_2^2$ (two-sided) | $F_o = s_M^2/s_m^2$<br>$s_M^2$=max($s_1^2$,$s_2^2$), $s_m^2$=min($s_1^2$,$s_2^2$) | $\nu_N = n_M$-1,$\nu_D = n_m$-1 |

(*) $n_M$ is the value of n corresponding to the $s_M$, and $n_m$ is the value of n corresponding to $s_m$.

The P-value is calculated, in all cases, as: P-value = P(F>$F_o$) = UTPF($\nu_N$, $\nu_D$,$F_o$)
The test criteria are:

- Reject $H_o$ if P-value < $\alpha$
- Do not reject $H_o$ if P-value > $\alpha$.

Example1 -- Consider two samples drawn from normal populations such that $n_1 = 21$, $n_2 = 31$, $s_1^2 = 0.36$, and $s_2^2 = 0.25$. We test the null hypothesis, $H_o$: $\sigma_1^2 = \sigma_2^2$, at a significance level $\alpha = 0.05$, against the alternative hypothesis, $H_1$: $\sigma_1^2 \neq \sigma_2^2$. For a two-sided hypothesis, we need to identify $s_M$ and $s_m$, as follows:

$$s_M^2=\max(s_1^2,s_2^2) = \max(0.36,0.25) = 0.36 = s_1^2$$
$$s_m^2=\min(s_1^2,s_2^2) = \max(0.36,0.25) = 0.25 = s_2^2$$

Also,

$$n_M = n_1 = 21,$$
$$n_m = n_2 = 31,$$
$$\nu_N = n_M - 1= 21\text{-}1=20,$$
$$\nu_D = n_m -1 = 31\text{-}1 =30.$$

Therefore, the F test statistics is $F_o = s_M^2/s_m^2=0.36/0.25=1.44$

The P-value is P-value = $P(F>F_o)$ = $P(F>1.44)$ = UTPF($\nu_N$, $\nu_D$,$F_o$) = UTPF(20,30,1.44) = 0.1788…

Since 0.1788… > 0.05, i.e., P-value > $\alpha$, therefore, we cannot reject the null hypothesis that $H_o$: $\sigma_1^2 = \sigma_2^2$.

# Additional notes on linear regression

In this section we elaborate the ideas of linear regression presented earlier in the chapter and present a procedure for hypothesis testing of regression parameters.

## The method of least squares

Let x = independent, non-random variable, and Y = dependent, random variable. The regression curve of Y on x is defined as the relationship between x and the mean of the corresponding distribution of the Y's.

Assume that the regression curve of Y on x is linear, i.e., mean distribution of Y's is given by $A + Bx$. Y differs from the mean ($A + B\cdot x$) by a value $\varepsilon$, thus $Y = A + B\cdot x + \varepsilon$, where $\varepsilon$ is a random variable.

To visually check whether the data follows a linear trend, draw a scattergram or scatter plot.

Suppose that we have n paired observations ($x_i$, $y_i$); we predict y by means of ^y = a + b·x, where a and b are constant.

Define the prediction error as, $e_i = y_i - \hat{}y_i = y_i - (a + b\cdot x_i)$.

The method of least squares requires us to choose a, b so as to minimize the sum of squared errors (SSE)

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} [y_i - (a + bx_i)]^2$$

the conditions

$$\frac{\partial}{\partial a}(SSE) = 0 \qquad \frac{\partial}{\partial b}(SSE) = 0$$

We get the, so-called, normal equations:

$$\sum_{i=1}^{n} y_i = a \cdot n + b \cdot \sum_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} x_i \cdot y_i = a \cdot \sum_{i=1}^{n} x_i + b \cdot \sum_{i=1}^{n} x_i^2$$

This is a system of linear equations with a and b as the unknowns, which can be solved using the linear equation features of the calculator. There is, however, no need to bother with these calculations because you can use the **3. Fit Data …** option in the ↱ STAT menu as presented earlier.

---

**Notes**:

- a,b are unbiased estimators of A, B.
- The Gauss-Markov theorem of probability indicates that among all unbiased estimators for A and B, the least-square estimators (a,b) are the most efficient.

---

## Additional equations for linear regression

The summary statistics such as $\Sigma x$, $\Sigma x^2$, etc., can be used to define the following quantities:

$$S_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 = (n-1)\cdot s_x^2 = \sum_{i=1}^{n} x_i^{\;2} - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)$$

$$S_y = \sum_{i=1}^{n}(y_i - \bar{y})^2 = (n-1)\cdot s_y^2 = \sum_{i=1}^{n} y_i^{\;2} - \frac{1}{n}\left(\sum_{i=1}^{n} y_i\right)^2$$

$$S_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})^2 = (n-1)\cdot s_{xy} = \sum_{i=1}^{n} x_i y_i - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)$$

From which it follows that the standard deviations of x and y, and the covariance of x,y are given, respectively, by

$$s_x = \sqrt{\frac{S_{xx}}{n-1}}\ ,\ s_y = \sqrt{\frac{S_{yy}}{n-1}}\text{, and }\ s_{xy} = \frac{S_{yx}}{n-1}$$

Also, the sample correlation coefficient is $r_{xy} = \dfrac{S_{xy}}{\sqrt{S_{xx}\cdot S_{yy}}}$.

In terms of $\bar{x}$, $\bar{y}$, $S_{xx}$, $S_{yy}$, and $S_{xy}$, the solution to the normal equations is:

$$a = \bar{y} - b\bar{x}\ ,\quad b = \frac{S_{xy}}{S_{xx}} = \frac{s_{xy}}{s_x^2}$$

## Prediction error

The regression curve of Y on x is defined as $Y = A + B\cdot x + \varepsilon$. If we have a set of n data points $(x_i, y_i)$, then we can write $Y_i = A + B\cdot x_i + \varepsilon_l$, $(i = 1,2,\ldots,n)$, where $Y_i$ = independent, normally distributed random variables with mean $(A + B\cdot x_i)$ and the common variance $\sigma^2$; $\varepsilon_i$ = independent, normally distributed random variables with mean zero and the common variance $\sigma^2$.

Let $y_i$ = actual data value, $\hat{}y_i = a + b\cdot x_i$ = least-square prediction of the data. Then, the prediction error is: $e_i = y_i - \hat{}y_i = y_i - (a + b\cdot x_i)$.

An estimate of $\sigma^2$ is the, so-called, standard error of the estimate,

$$s_e^2 = \frac{1}{n-2}\sum_{i=1}^{n}[y_i - (a+bx_i)]^2 = \frac{S_{yy} - (S_{xy})^2/S_{xx}}{n-2} = \frac{n-1}{n-2}\cdot s_y^2 \cdot (1-r_{xy}^2)$$

## Confidence intervals and hypothesis testing in linear regression

Here are some concepts and equations related to statistical inference for linear regression:

- Confidence limits for regression coefficients:
  For the slope (B): $b - (t_{n-2,\alpha/2})\cdot s_e/\sqrt{S_{xx}} < B < b + (t_{n-2,\alpha/2})\cdot s_e/\sqrt{S_{xx}}$,
  For the intercept (A):
  $a - (t_{n-2,\alpha/2})\cdot s_e\cdot[(1/n)+ \bar{x}^2/S_{xx}]^{1/2} < A <$
  $a + (t_{n-2,\alpha/2})\cdot s_e\cdot[(1/n)+ \bar{x}^2/S_{xx}]^{1/2}$,
  where t follows the Student's t distribution with $\nu = n - 2$, degrees of freedom, and n represents the number of points in the sample.

- Hypothesis testing on the slope, B:
  Null hypothesis, $H_0$: $B = B_0$, tested against the alternative hypothesis, $H_1$: $B \neq B_0$. The test statistic is $t_0 = (b - B_0)/(s_e/\sqrt{S_{xx}})$, where t follows the Student's t distribution with $\nu = n - 2$, degrees of freedom, and n represents the number of points in the sample. The test is carried out as that of a mean value hypothesis testing, i.e., given the level of significance, $\alpha$, determine the critical value of t, $t_{\alpha/2}$, then, reject $H_0$ if $t_0 > t_{\alpha/2}$ or if $t_0 < - t_{\alpha/2}$.

  If you test for the value $B_0$= 0, and it turns out that the test suggests that you do not reject the null hypothesis, $H_0$: $B = 0$, then, the validity of a linear regression is in doubt. In other words, the sample data does not support the assertion that $B \neq 0$. Therefore, this is a test of the significance of the regression model.

- Hypothesis testing on the intercept , A:
  Null hypothesis, $H_0$: A = $A_0$, tested against the alternative hypothesis, $H_1$: $A \neq A_0$. The test statistic is $t_0 = (a\text{-}A_0)/[(1/n)+ \bar{x}^2/S_{xx}]^{1/2}$, where t follows the Student's t distribution with $\nu$ = n – 2, degrees of freedom, and n represents the number of points in the sample. The test is carried out as that of a mean value hypothesis testing, i.e., given the level of significance, $\alpha$, determine the critical value of t, $t_{\alpha/2}$, then, reject $H_0$ if $t_0 > t_{\alpha/2}$ or if $t_0 < - t_{\alpha/2}$.

- Confidence interval for the mean value of Y at x = $x_0$, i.e., $\alpha+\beta x_0$:
  $a+b\cdot x-(t_{n\text{-}2,\alpha/2})\cdot s_e\cdot[(1/n)+(x_0\text{-} \bar{x})^2/S_{xx}]^{1/2} < \alpha+\beta x_0 <$
  $a+b\cdot x+(t_{n\text{-}2, \alpha /2})\cdot s_e\cdot[(1/n)+(x_0\text{-} \bar{x})^2/S_{xx}]^{1/2}$.

- Limits of prediction: confidence interval for the predicted value $Y_0$=$Y(x_0)$:
  $a+b\cdot x-(t_{n\text{-}2,\alpha/2})\cdot s_e\cdot[1+(1/n)+(x_0\text{-} \bar{x})^2/S_{xx}]^{1/2} < Y_0 <$
  $a+b\cdot x+(t_{n\text{-}2, \alpha /2})\cdot s_e\cdot[1+(1/n)+(x_0\text{-} \bar{x})^2/S_{xx}]^{1/2}$.

## Procedure for inference statistics for linear regression using the calculator

1) Enter (x,y) as columns of data in the statistical matrix ΣDAT.
2) Produce a scatterplot for the appropriate columns of ΣDAT, and use appropriate H- and V-VIEWS to check linear trend.
3) Use ⮣ STAT ▼ ▼ OK, to fit straight line, and get a, b, $s_{xy}$ (Covariance), and $r_{xy}$ (Correlation).
4) Use ⮣ STAT ▼ OK, to obtain $\bar{x}$, $\bar{y}$, $s_x$, $s_y$. Column 1 will show the statistics for x while column 2 will show the statistics for y.
5) Calculate

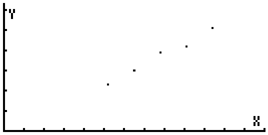
$$S_{xx} = (n-1)\cdot s_x^2, \quad s_e^2 = \frac{n-1}{n-2}\cdot s_y^2\cdot(1-r_{xy}^2)$$

6) For either confidence intervals or two-tailed tests, obtain $t_{\alpha/2}$, with (1-$\alpha$)100% confidence, from t-distribution with $\nu$ = n -2.
7) For one- or two-tailed tests, find the value of t using the appropriate equation for either A or B. Reject the null hypothesis if `P-value` < $\alpha$.
8) For confidence intervals use the appropriate formulas as shown above.

Example 1 -- For the following (x,y) data, determine the 95% confidence interval for the slope B and the intercept A

| **x** | 2.0 | 2.5 | 3.0 | 3.5 | 4.0 |
|---|---|---|---|---|---|
| **y** | 5.5 | 7.2 | 9.4 | 10.0 | 12.2 |

Enter the (x,y) data in columns 1 and 2 of ΣDAT, respectively. A scatterplot of the data shows a good linear trend:



Use the `Fit Data..` option in the ↱ STAT menu, to get:

```
3: '-.86 + 3.24*X'
2: Correlation: 0.989720229749
1: Covariance: 2.025
```

These results are interpreted as a = -0.86, b = 3.24, $r_{xy}$ = 0.989720229749, and $s_{xy}$ = 2.025. The correlation coefficient is close enough to 1.0 to confirm the linear trend observed in the graph.

From the `Single-var…` option of the ↱ STAT menu we find: $\overline{x}$ = 3, $s_x$ = 0.790569415042, $\overline{y}$ = 8.86, $s_y$ = 2.58804945857.

Next, with n = 5, calculate

$$S_{xx} = (n-1)\cdot s_x^2 = (5-1)\cdot 0.790569415042^2 = 2.5$$

$$s_e^2 = \frac{n-1}{n-2}\cdot s_y^2 \cdot (1-r_{xy}^2) =$$

$$\frac{5-1}{5-2}\cdot 2.5880...^2 \cdot (1-0.9897...^2) = 0.1826...$$

Confidence intervals for the slope (B) and intercept (A):

- First, we obtain $t_{n-2,\alpha/2} = t_{3,0.025}$ = 3.18244630528 (See chapter 17 for a program to solve for $t_{\nu,a}$):
- Next, we calculate the terms

$$(t_{n-2,\alpha/2})\cdot s_e/\sqrt{S_{xx}} = 3.182\ldots\cdot(0.1826\ldots/2.5)^{1/2} = 0.8602\ldots$$

$$(t_{n-2,\alpha/2})\cdot s_e\cdot[(1/n)+ \bar{x}^2/S_{xx}]^{1/2} =$$
$$3.1824\ldots\cdot\sqrt{0.1826\ldots}\cdot[(1/5)+3^2/2.5]^{1/2} = 2.65$$

- Finally, for the slope B, the 95% confidence interval is

(-0.86-0.860242, -0.86+0.860242) = (-1.72, -0.00024217)

For the intercept A, the 95% confidence interval is (3.24-2.6514, 3.24+2.6514) = (0.58855,5.8914).

Example 2 -- Suppose that the y-data used in Example 1 represent the elongation (in hundredths of an inch) of a metal wire when subjected to a force x (in tens of pounds). The physical phenomenon is such that we expect the intercept, A, to be zero. To check if that should be the case, we test the null hypothesis, $H_0$: $A = 0$, against the alternative hypothesis, $H_1$: $A \neq 0$, at the level of significance $\alpha = 0.05$.

The test statistic is $t_0 = (a-0)/[(1/n)+ \bar{x}^2/S_{xx}]^{1/2} = (-0.86)/ [(1/5)+3^2/2.5]^{½} = -0.44117$. The critical value of t, for $\nu = n - 2 = 3$, and $\alpha/2 = 0.025$, can be calculated using the numerical solver for the equation $\alpha$ = UTPT($\gamma$,t) developed in Chapter 17. In this program, $\gamma$ represents the degrees of freedom (n-2), and $\alpha$ represents the probability of exceeding a certain value of t, i.e., Pr[ $t>t_\alpha$] = 1 – $\alpha$. For the present example, the value of the level of significance is $\alpha$ = 0.05, g = 3, and $t_{n-2,\alpha/2} = t_{3,0.025}$. Also, for $\gamma$ = 3 and $\alpha$ = 0.025, $t_{n-2,\alpha/2} = t_{3,0.025}$ = 3.18244630528. Because $t_0 > - t_{n-2,\alpha/2}$, we cannot reject the null hypothesis, $H_0$: $A = 0$, against the alternative hypothesis, $H_1$: $A \neq 0$, at the level of significance $\alpha$ = 0.05.

This result suggests that taking A = 0 for this linear regression should be acceptable. After all, the value we found for a, was –0.86, which is relatively close to zero.

Example 3 – Test of significance for the linear regression. Test the null hypothesis for the slope $H_0$: $\mathrm{B} = 0$, against the alternative hypothesis, $H_1$: $\mathrm{B} \neq 0$, at the level of significance $\alpha = 0.05$, for the linear fitting of Example 1.

The test statistic is $t_0 = (b - \mathrm{B}_0)/(s_e/\sqrt{S_{xx}}) = (3.24\text{-}0)/(\sqrt{0.18266666667}/2.5) = 18.95$. The critical value of t, for $\nu = n - 2 = 3$, and $\alpha/2 = 0.025$, was obtained in Example 2, as $t_{n\text{-}2,\alpha/2} = t_{3,0.025} = 3.18244630528$. Because, $t_0 > t_{\alpha/2}$, we must reject the null hypothesis $H_1$: $\mathrm{B} \neq 0$, at the level of significance $\alpha = 0.05$, for the linear fitting of Example 1.

## Multiple linear fitting

Consider a data set of the form

| $x_1$ | $x_2$ | $x_3$ | … | $x_n$ | y |
|---|---|---|---|---|---|
| $x_{11}$ | $x_{21}$ | $x_{31}$ | … | $x_{n1}$ | $y_1$ |
| $x_{12}$ | $x_{22}$ | $x_{32}$ | … | $x_{n2}$ | $y_2$ |
| $x_{13}$ | $x_{32}$ | $x_{33}$ | … | $x_{n3}$ | $y_3$ |
| . | . | . |  | . | . |
| . | . | . | . | . | . |
| $x_{1,m\text{-}1}$ | $x_{2,m\text{-}1}$ | $x_{3,m\text{-}1}$ | … | $x_{n,m\text{-}1}$ | $y_{m\text{-}1}$ |
| $x_{1,m}$ | $x_{2,m}$ | $x_{3,m}$ | … | $x_{n,m}$ | $y_m$ |

Suppose that we search for a data fitting of the form $y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + b_3 \cdot x_3 + \ldots + b_n \cdot x_n$. You can obtain the least-square approximation to the values of the coefficients $\mathbf{b} = [b_0 \;\; b_1 \;\; b_2 \;\; b_3 \ldots b_n]$, by putting together the matrix **X:**

$$\begin{bmatrix} 1 & x_{11} & x_{21} & x_{31} & \cdots & x_{n1} \\ 1 & x_{12} & x_{22} & x_{32} & \cdots & x_{n2} \\ 1 & x_{13} & x_{32} & x_{33} & \cdots & x_{n3} \\ . & . & . & . & & . \\ . & . & . & . & . & . \\ 1 & x_{1,m} & x_{2,m} & x_{3,m} & \cdots & x_{n,m} \end{bmatrix}$$

Then, the vector of coefficients is obtained from $\mathbf{b} = (\mathbf{X}^T\cdot\mathbf{X})^{-1}\cdot\mathbf{X}^T\cdot\mathbf{y}$, where **y** is the vector $\mathbf{y} = [y_1\ y_2\ \ldots\ y_m]^T$.

For *example*, use the following data to obtain the multiple linear fitting

$$y = b_0 + b_1\cdot x_1 + b_2\cdot x_2 + b_3\cdot x_{3,}$$

| $x_1$ | $x_2$ | $x_3$ | $y$ |
|---|---|---|---|
| 1.20 | 3.10 | 2.00 | 5.70 |
| 2.50 | 3.10 | 2.50 | 8.20 |
| 3.50 | 4.50 | 2.50 | 5.00 |
| 4.00 | 4.50 | 3.00 | 8.20 |
| 6.00 | 5.00 | 3.50 | 9.50 |

With the calculator, in RPN mode, you can proceed as follows:

First, within your HOME directory, create a sub-directory to be called MPFIT (Multiple linear and Polynomial data FITting) , and enter the MPFIT sub-directory. Within the sub-directory, type this program:

```
« → X y « X TRAN X * INV X TRAN * y * » »
```

and store it in a variable called MTREG (MulTiple REGression).

Next, enter the matrices **X** and **b** into the stack:

[[1,1.2,3.1,2][1,2.5,3.1,2.5 ][1,3.5,4.5,2.5][1,4,4.5,3][1,6,5,3.5]]

ENTER ENTER (keep an extra copy)

[5.7,8.2,5.0,8.2,9.5] ENTER

Press VAR MTREG. The result is: [-2.1649…,–0.7144…,-1.7850…,7.0941…], i.e.,

$$y = -2.1649–0.7144\cdot x_1 -1.7850\times10^{-2}\cdot x_2 + 7.0941\cdot x_3 .$$

You should have in your calculator's stack the value of the matrix X and the vector b, the fitted values of y are obtained from **y** = **X**·**b**, thus, just press × to obtain: [5.63.., 8.25.., 5.03.., 8.23.., 9.45..].

Compare these fitted values with the original data as shown in the table below:

| $x_1$ | $x_2$ | $x_3$ | y | y-fitted |
|---|---|---|---|---|
| 1.20 | 3.10 | 2.00 | 5.70 | 5.63 |
| 2.50 | 3.10 | 2.50 | 8.20 | 8.25 |
| 3.50 | 4.50 | 2.50 | 5.00 | 5.03 |
| 4.00 | 4.50 | 3.00 | 8.20 | 8.23 |
| 6.00 | 5.00 | 3.50 | 9.50 | 9.45 |

## Polynomial fitting

Consider the x-y data set {($x_1$,$y_1$), ($x_2$,$y_2$), …, ($x_n$,$y_n$)}. Suppose that we want to fit a polynomial or order p to this data set. In other words, we seek a fitting of the form $y = b_0 + b_1 \cdot x + b_2 \cdot x^2 + b_3 \cdot x^3 + \ldots + b_p \cdot x^p$. You can obtain the least-square approximation to the values of the coefficients **b** = [$b_0$ $b_1$ $b_2$ $b_3$ … $b_p$], by putting together the matrix **X**

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \ldots & x_1^{p-1} & y_1^{\,p} \\ 1 & x_2 & x_2^2 & x_2^3 & \ldots & x_2^{\,p-1} & y_2^{\,p} \\ 1 & x_3 & x_3^2 & x_3^3 & \ldots & x_3^{\,p-1} & y_3^{\,p} \\ . & . & . & . & & . & . \\ . & . & . & . & . & . & . \\ 1 & x_n & x_n^{\,2} & x_n^3 & \ldots & x_n^{\,p-1} & y_n^{\,p} \end{bmatrix}$$

Then, the vector of coefficients is obtained from $\mathbf{b} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$, where **y** is the vector $\mathbf{y} = [y_1\ y_2\ \ldots\ y_n]^T$.

In Chapter 10, we defined the Vandermonde matrix corresponding to a vector $\mathbf{x} = [x_1\ x_2\ \ldots\ x_m]$ . The Vandermonde matrix is similar to the matrix **X** of interest to the polynomial fitting, but having only *n*, rather than (*p+1*) columns.

We can take advantage of the VANDERMONDE function to create the matrix **X** if we observe the following rules:

If $p = n\text{-}1$, $\mathbf{X} = \mathbf{V}_n$.
If $p < n\text{-}1$, then remove columns $p+2, \ldots, n\text{-}1, n$ from $\mathbf{V}_n$ to form $\mathbf{X}$.
If $p > n\text{-}1$, then add columns $n+1, \ldots, p\text{-}1, p+1$, to $\mathbf{V}_n$ to form matrix $\mathbf{X}$.

In step 3 from this list, we have to be aware that column $i$ ($i= n+1, n+2, \ldots, p+1$) is the vector $[x_1^i\ x_2^i \ldots x_n^i]$. If we were to use a list of data values for x rather than a vector, i.e., $\mathbf{x} = \{ x_1\ x_2 \ldots x_n \}$, we can easily calculate the sequence $\{ x_1^i\ x_2^i \ldots x_n^i \}$. Then, we can transform this list into a vector and use the COL menu to add those columns to the matrix $\mathbf{V}_n$ until $\mathbf{X}$ is completed.

After **X** is ready, and having the vector **y** available, the calculation of the coefficient vector **b** is the same as in multiple linear fitting (the previous matrix application). Thus, we can write a program to calculate the polynomial fitting that can take advantage of the program already developed for multiple linear fitting. We need to add to this program the steps 1 through 3 listed above.

The *algorithm* for the program, therefore, can be written as follows:

Enter vectors **x** and **y,** of the same dimension, as lists. (Note: since the function VANDERMONDE uses a list as input, it is more convenient to enter the (x,y) data as a list.) Also, enter the value of p.

```
•  Determine n = size of vector x.
•  Use the function VANDERMONDE to generate the Vandermonde
   matrix $V_n$ for the list x entered.
•  If $p = n-1$, then
       $X = V_n$,
   Else If  $p < n-1$
           Remove columns $p+2, …, n$ from $V_n$ to form X
           (Use a FOR loop and COL-)
   Else
           Add columns $n+1, …, p+1$ to $V_n$ to form X
           (FOR loop, calculate $x^i$, convert to vector, use COL+)
•  Convert y to vector
•  Calculate b using program MTREG (see example on multiple linear
   fitting above)
```

Here is the *translation of the algorithm* to a program in User RPL language. (See Chapter 21 for additional information on programming):

```
«                              Open program
 → x y p                       Enter lists x and y, and p (levels 3,2,1)
 «                             Open subprogram 1
 x SIZE → n                    Determine size of x list
 «                             Open subprogram 2
   x VANDERMONDE               Place x in stack, obtain Vn
   IF 'p<n-1' THEN             This IF implements step 3 in algorithm
     n                         Place n in stack
     p 2 +                     Calculate p+1
     FOR j                     Start loop j = n-1, n-2, …, p+1, step = -1
         j  COL– DROP          Remove column and drop it from stack
    -1 STEP                    Close FOR-STEP loop
   ELSE
     IF 'p>n-1' THEN
       n 1 +                   Calculate n+1
       p 1 +                   Calculate p+1
      FOR j                    Start a loop with  j = n, n+1, …, p+1.
        x  j  ^                Calculate xj, as a list
        OBJ→ →ARRY             Convert list to array
        j COL+                 Add column to matrix
      NEXT                     Close FOR-NEXT loop
    END                        Ends second IF clause.
 END                           Ends first IF clause.  Its result is X
 y  OBJ→ →ARRY                 Convert list y to an array
 MTREG                         X and y used by program MTREG
 →NUM                          Convert to decimal format
 »                             Close sub-program 2
 »                             Close sub-program 1
»                              Close main program
```

Save it into a variable called POLY (POLYnomial fitting).

As an *example*, use the following data to obtain a polynomial fitting with p = 2, 3, 4, 5, 6.

| x | y |
|---|---|
| 2.30 | 179.72 |
| 3.20 | 562.30 |
| 4.50 | 1969.11 |
| 1.65 | 65.87 |
| 9.32 | 31220.89 |
| 1.18 | 32.81 |
| 6.24 | 6731.48 |
| 3.45 | 737.41 |
| 9.89 | 39248.46 |
| 1.22 | 33.45 |

Because we will be using the same x-y data for fitting polynomials of different orders, it is advisable to save the lists of data values x and y into variables xx and yy, respectively. This way, we will not have to type them all over again in each application of the program POLY. Thus, proceed as follows:

{ 2.3 3.2 4.5 1.65 9.32 1.18 6.24 3.45 9.89 1.22 } ENTER 'xx' STO▶
{179.72 562.30 1969.11 65.87 31220.89 32.81 6731.48 737.41 39248.46 33.45} ENTER 'yy' STO▶

To fit the data to polynomials use the following:

XX YY 2 POLY, Result: [4527.73 -3958.52 742.23]
i.e., $y = 4527.73\text{-}39.58x+742.23x^2$
XX YY 3 POLY, Result: [ –998.05 1303.21 -505.27 79.23]
i.e., $y = -998.05+1303.21x\text{-}505.27x^2+79.23x^3$
XX YY 4 POLY, Result: [20.92 –2.61 –1.52 6.05 3.51 ]
i.e., $= 20.92\text{-}2.61x\text{-}1.52x^2+6.05x^3+3.51x^4.$
XX YY 5 POLY, Result: [19.08 0.18 –2.94 6.36 3.48 0.00 ]
i.e., $y = 19.08+0.18x\text{-}2.94x^2+6.36x^3+3.48x^4+0.0011x^5$
XX YY 6 POLY, Result: [-16.73 67.17 –48.69 21.11 1.07 0.19 0.00]
i.e., $y = -16.73+67.17x\text{-}48.69x^2+21.11x^3+1.07x^4+0.19x^5+0.0058x^6$

## Selecting the best fitting

As you can see from the results above, you can fit any polynomial to a set of data. The question arises, which is the best fitting for the data? To help one decide on the best fitting we can use several criteria:

- The correlation coefficient, r. This value is constrained to the range $-1 < r < 1$. The closer *r* is to +1 or –1, the better the data fitting.
- The sum of squared errors, SSE. This is the quantity that is to be minimized by least-square approach.
- A plot of residuals. This is a plot of the error corresponding to each of the original data points. If these errors are completely random, the residuals plot should show no particular trend.

Before attempting to program these criteria, we present some definitions:

Given the vectors **x** and **y** of data to be fit to the polynomial equation, we form the matrix **X** and use it to calculate a vector of polynomial coefficients **b**. We can calculate a *vector of fitted data*, **y**', by using $\mathbf{y}' = \mathbf{X}\cdot\mathbf{b}$.

An *error vector* is calculated by $\mathbf{e} = \mathbf{y} - \mathbf{y}'$.

The *sum of square errors* is equal to the square of the magnitude of the error vector, i.e., $SSE = |\mathbf{e}|^2 = \mathbf{e}\bullet\mathbf{e} = \Sigma\ e_i^2 = \Sigma\ (y_i\text{-}y'_i)^2$.

To calculate the correlation coefficient we need to calculate first what is known as the *sum of squared totals*, SST, defined as $SST = \Sigma\ (y_i\text{-}\ \overline{y})^2$, where $\overline{y}$ is the *mean value* of the original y values, i.e., $\overline{y} = (\Sigma y_i)/n$.

In terms of SSE and SST, the correlation coefficient is defined by

$$r = [1\text{-}(SSE/SST)]^{1/2}\ .$$

Here is the new program including calculation of SSE and r (Once more, consult the last page of this chapter to see how to produce the variable and command names in the program):

```
«                                 Open program
 → x y p                          Enter lists x and y, and number p
 «                                Open subprogram1
 x SIZE → n                       Determine size of x list
  «                               Open subprogram 2
   x VANDERMONDE                  Place x in stack, obtain Vn
   IF 'p<n-1' THEN                This IF is step 3 in algorithm
     n                            Place n in stack
     p 2 +                        Calculate p+1
     FOR j                        Start loop, j = n-1 to p+1, step = -1
        j  COL– DROP              Remove column, drop from stack
    -1 STEP                       Close FOR-STEP loop
  ELSE
     IF 'p>n-1' THEN
       n 1 +                      Calculate n+1
       p 1 +                      Calculate p+1
      FOR j                       Start loop with  j = n, n+1, …, p+1.
         x  j  ^                  Calculate xʲ, as a list
         OBJ→ →ARRY               Convert list to array
         j   COL+                 Add column to matrix
      NEXT                        Close FOR-NEXT loop
    END                           Ends second IF clause.
 END                              Ends first IF clause.  Produces X
 y  OBJ→   →ARRY                  Convert list y to an array
 →    X   yv                      Enter matrix and array as X and y
   «                              Open subprogram 3
    X   yv   MTREG                X and y used by program MTREG
    →NUM                          If needed, converts to floating point
    → b                           Resulting vector passed as b
     «                            Open subprogram 4
      b  yv                       Place b and yv in stack
      X b *                       Calculate X·b
       -                          Calculate e = y - X·b
      ABS  SQ DUP                 Calculate SSE, make copy
      y  ΣLIST n /                Calculate  ȳ
      n 1 →LIST SWAP CON          Create vector of n values of  ȳ
```

```
        yv – ABS SQ                      Calculate SST
        /                                Calculate SSE/SST
        NEG 1 + √                        Calculate r = [1–SSE/SST ]^1/2
        "r" →TAG                         Tag result as "r"
        SWAP                             Exchange stack levels 1 and 2
       "SSE" →TAG                        Tag result as SSE
      »                                  Close sub-program 4
     »                                   Close sub-program 3
    »                                    Close sub-program 2
   »                                     Close sub-program 1
  »                                      Close main program
```

Save this program under the name POLYR, to emphasize calculation of the correlation coefficient r.

Using the POLYR program for values of p between 2 and 6 produce the following table of values of the correlation coefficient, r, and the sum of square errors, SSE:

| *p* | *r* | *SSE* |
|---|---|---|
| 2 | 0.9971908 | 10731140.01 |
| 3 | 0.9999768 | 88619.36 |
| 4 | 0.9999999 | 7.48 |
| 5 | 0.9999999 | 8.92 |
| 6 | 0.9999998 | 432.61 |

While the correlation coefficient is very close to 1.0 for all values of p in the table, the values of SSE vary widely. The smallest value of SSE corresponds to p = 4. Thus, you could select the preferred polynomial data fitting for the original x-y data as:

$$y = 20.92-2.61x-1.52x^2+6.05x^3+3.51x^4.$$

# Chapter 19
# Numbers in Different Bases

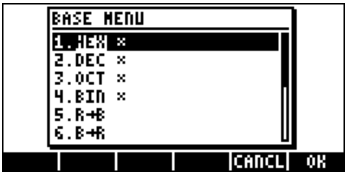
In this Chapter we present examples of calculations of number in bases other than the decimal basis.

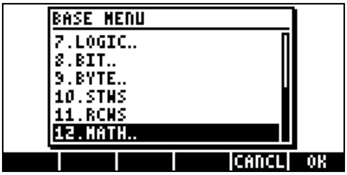
## Definitions

The number system used for everyday arithmetic is known as the *decimal* system for it uses 10 (Latin, deca) digits, namely 0-9, to write out any real number. Computers, on the other hand, use a system that is based on two possible states, or *binary* system. These two states are represented by 0 and 1, ON and OFF, or high-voltage and low-voltage. Computers also use number systems based on eight digits (0-7) or *octal* system, and sixteen digits (0-9, A-F) or *hexadecimal*. As in the decimal system, the relative position of digits determines its value. In general, a number n in base b can be written as a series of digits n = $(a_1a_2 \ldots a_n.c_1c_2 \ldots c_m)_b$. The “point” separates *n* “integer” digits from m “decimal” digits. The value of the number, converted to our customary decimal system, is calculated by using $n = a_1 \cdot bn^{-1} + a_2 \cdot b^{n-2} + \ldots + a_nb^0 + c_1 \cdot b^{-1} + c_2 \cdot b^{-2} + \ldots + c_m \cdot b^{-m}$. For example, $(15.234)_{10} = 1 \cdot 10^1 + 5 \cdot 10^0 + 2 \cdot 10^{-1} + 3 \cdot 10^{-2} + 4 \cdot 10^{-3}$, and $(101.111)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$

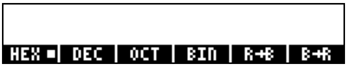
## The BASE menu

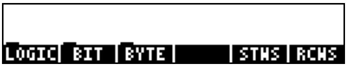
While the calculator would typically be operated using the decimal system, you can produce calculations using the binary, octal, or hexadecimal system. Many of the functions for manipulating number systems other than the decimal system are available in the BASE menu, accessible through ↱ BASE (the 3 key). With system flag 117 set to CHOOSE boxes, the BASE menu shows the following entries:





With system flag 117 set to SOFT menus, the BASE menu shows the following:





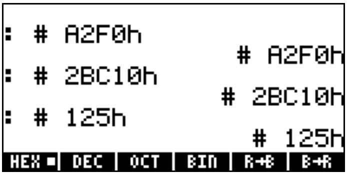
With this format, it is evident that the LOGIC, BIT, and BYTE entries within the BASE menu are themselves sub-menus. These menus are discussed later in this Chapter.

## Functions HEX, DEC, OCT, and BIN

Numbers in non-decimal systems are written preceded by the # symbol in the calculator. The symbol # is readily available as ← # (the 3 key). To select which number system (current base) will be used for numbers preceded by #, select one of the following functions in the first BASE menu, i.e., HEX(adecimal), DEC(imal), OCT(al), or BIN(ary). For example, if HEX■ is selected, any number written in the calculator that starts with # will be a hexadecimal number. Thus, you can write numbers such as #53, #A5B, etc. in this system. As different systems are selected, the numbers will be automatically converted to the new current base.

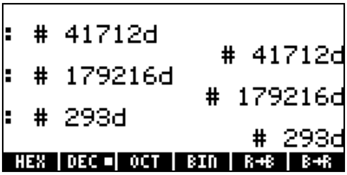
The following examples show the same three numbers written with the # symbol for different current bases:

HEX



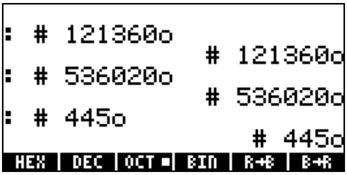
```
: # A2F0h
                # A2F0h
: # 2BC10h
               # 2BC10h
: # 125h
                 # 125h
```

DEC



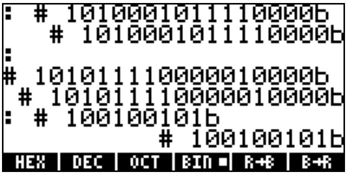
```
: # 41712d
                # 41712d
: # 179216d
               # 179216d
: # 293d
                  # 293d
```

OCT



```
: # 121360o
               # 121360o
: # 536020o
               # 536020o
: # 445o
                  # 445o
```

BIN

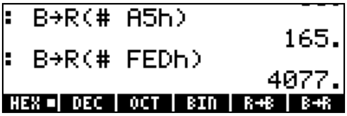


```
: # 1010001011110000b
   # 1010001011110000b
:
# 101011110000010000b
 # 101011110000010000b
: # 100100101b
          # 100100101b
```

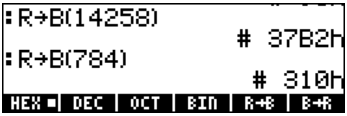
As the decimal (DEC) system has 10 digits (0,1,2,3,4,5,6,7,8,9), the hexadecimal (HEX) system has 16 digits (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F), the octal (OCT) system has 8 digits (0,1,2,3,4,5,6,7), and the binary (BIN) system has only 2 digits (0,1).

## Conversion between number systems

Whatever the number system selected, it is referred to as the <u>binary system</u> for the purpose of using the functions R→B and B→R. For example, if HEX■ is selected, the function B→R will convert any hexadecimal number (preceded by #) into a decimal number, while the function R→B works in the opposite direction. Try the following exercises, HEX is the current base:

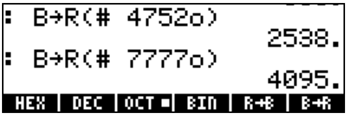


```
: B→R(# A5h)
                165.
: B→R(# FEDh)
               4077.
HEX■ | DEC | OCT | BIN | R→B | B→R
```

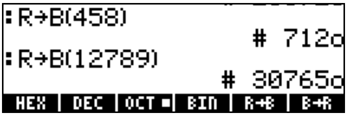


```
:R→B(14258)
            # 37B2h
:R→B(784)
             # 310h
HEX■ | DEC | OCT | BIN | R→B | B→R
```

The following examples show conversions when the base is the octal system:

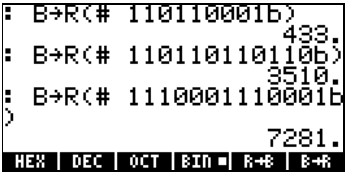


```
: B→R(# 4752o)
               2538.
: B→R(# 7777o)
               4095.
HEX | DEC | OCT■ | BIN | R→B | B→R
```

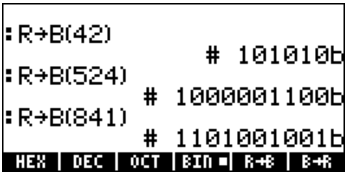


```
:R→B(458)
             # 712o
:R→B(12789)
           # 30765o
HEX | DEC | OCT■ | BIN | R→B | B→R
```

We also present transformations using the binary system as the current base:



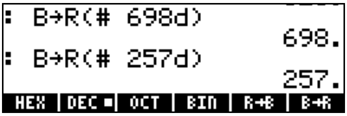
```
: B→R(# 110110001b)
                433.
: B→R(# 110110110110b)
               3510.
: B→R(# 1110001110001b
)
               7281.
HEX | DEC | OCT | BIN■ | R→B | B→R
```



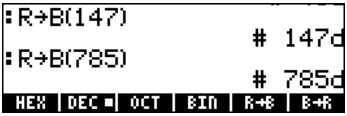
```
:R→B(42)
          # 101010b
:R→B(524)
      # 1000001100b
:R→B(841)
      # 1101001001b
HEX | DEC | OCT | BIN■ | R→B | B→R
```

Notice that every time you enter a number starting with #, you get as the entry the number you entered preceded by # and followed by the letter h, o, or b (hexadecimal, octal, or binary). The type of letter used as suffix depends on which non-decimal number system has been selected, i.e., HEX, OCT, or BIN.

To see what happens if you select the DEC setting, try the following conversions:



```
: B→R(# 698d)
                698.
: B→R(# 257d)
                257.
```



```
:R→B(147)
              # 147d
:R→B(785)
              # 785d
```

The only effect of selecting the DECimal system is that decimal numbers, when started with the symbol #, are written with the suffix d.

## Wordsize

The wordsize is the number of bits in a binary object. By default, the wordsize is 64 bites. Function RCWS (ReCall WordSize) shows the current wordsize. Function STWS (SeT the WordSize) allows the user to reset the wordsize to any number between 0 and 64.

Changing the wordsize will affect the way that binary integer operations are performed. For example, if a binary integer exceeds the current wordsize, the leading bits will be dropped before any operation can be performed on such number.

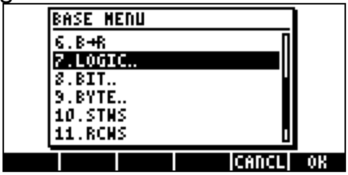
## Operations with binary integers

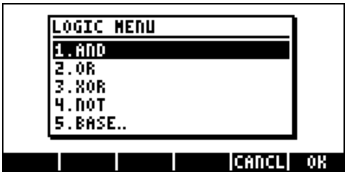
The operations of addition, subtraction, change of sign, multiplication, and division are defined for binary integers. Some examples, of addition and subtraction, are shown below, for different current bases:

```
           #A02h + #12Ah = #B2Ch
          #2562d + #298d = #2860d
          #5002o + #452o = #5454o
#101000000010b + #100101010b = #101100101100b

           #A02h - #12Ah = #8D8h
          #2562d - #298d = #2264d
          #5002o - #452o = #4330o
#101000000010b - #100101010b = #100011011000b
```

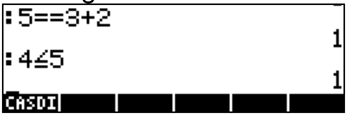
## The LOGIC menu

The LOGIC menu, available through the BASE (↱ BASE) provides the following functions:

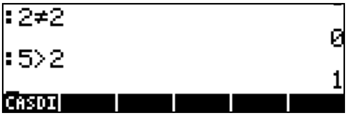




The functions AND, OR, XOR (exclusive OR), and NOT are logical functions. The input to these functions are two values or expressions (one in the case of NOT) that can be expressed as binary logical results, i.e., 0 or 1. Comparisons of numbers through the comparison operators =, ≠, >, <, ≤, and ≥, are logical statements that can be either true (1) or false (0). Some examples of logical statements are shown below:



```
:5==3+2
                    1
:4≤5
                    1
```



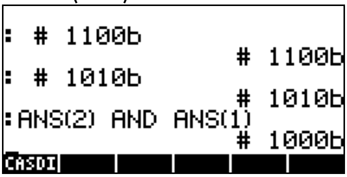
```
:2≠2
                    0
:5>2
                    1
```

Functions AND, OR, XOR, and NOT can be applied to comparison statements under the following rules:

| | | | |
|---|---|---|---|
| 1 AND 1 = 1 | 1 AND 0 = 0 | 0 AND 1 = 0 | 0 AND 0 = 0 |
| 1 OR 1 = 1 | 1 OR 0 = 1 | 0 OR 1 = 1 | 0 OR 0 = 0 |
| 1 XOR 1 = 0 | 1 XOR 0 = 1 | 0 XOR 1 = 1 | 0 XOR 0 = 0 |
| NOT(1) = 0 | NOT(0) = 1 | | |

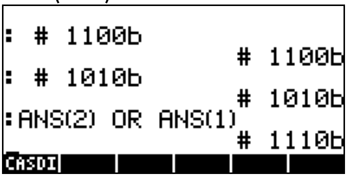
These functions can be used to build logical statements for programming purposes. In the context of this Chapter, they will by used to provide the result of bit-by-bit operations along the lines of the rules provided above. In the following examples, the base number system is indicated in parentheses:

AND (BIN)



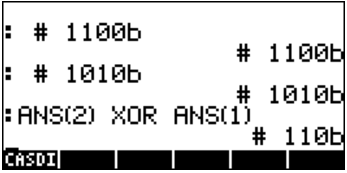
```
: # 1100b
                  # 1100b
: # 1010b
                  # 1010b
:ANS(2) AND ANS(1)
                  # 1000b
```

OR (BIN)



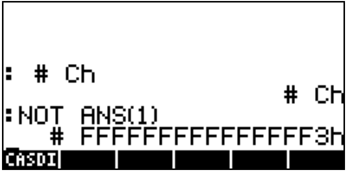
```
: # 1100b
                  # 1100b
: # 1010b
                  # 1010b
:ANS(2) OR ANS(1)
                  # 1110b
```

XOR (BIN)



```
: # 1100b
                    # 1100b
: # 1010b
                    # 1010b
:ANS(2) XOR ANS(1)
                     # 110b
```

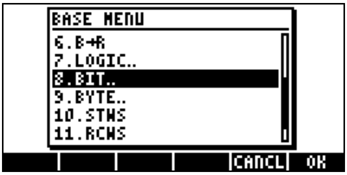
NOT (HEX)

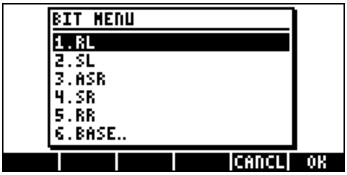


```
: # Ch
                       # Ch
:NOT ANS(1)
   # FFFFFFFFFFFFFFF3h
```

## The BIT menu

The BIT menu, available through the BASE (↱ BASE ) provides the following functions:



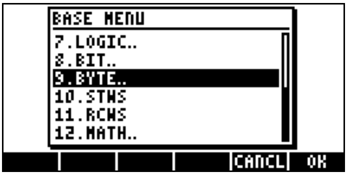


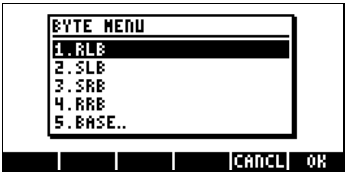
Functions RL, SL, ASR, SR, RR, contained in the BIT menu, are used to manipulate bits in a binary integer.  The definition of these functions are shown below:

RL:  Rotate Left one bit, e.g., #1100b → #1001b
SL:   Shift Left one bit, e.g., #1101b → #11010b
ASR: Arithmetic Shift Right one bit, e.g., #1100010b → #110001b
SR:   Shift Right one bit, e.g., #11011b →#1101b
RR:   Rotate Right one bit, e.g., #1101b → #1110b

## The  BYTE menu

The BYTE menu, available through the BASE (↱ BASE ) provides the following functions:





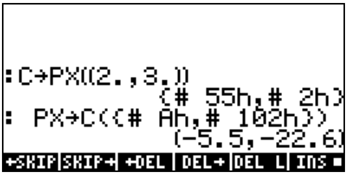
Functions RLB, SLB, SRB, RRB, contained in the BIT menu, are used to manipulate bits in a binary integer. The definition of these functions are shown below:

RLB: Rotate Left one byte, e.g., #1100b → #1001b
SLB: Shift Left one byte, e.g., #1101b → #11010b
SRB: Shift Right one byte, e.g., #11011b →#1101b
RRB: Rotate Right one byte, e.g., #1101b → #1110b

## Hexadecimal numbers for pixel references

Many plot option specifications use pixel references as input, e.g., { #332h #A23h } #Ah 0. 360. ARC, to draw an arc of a circle. We use functions C→PX and PX→C to convert quickly between user-unit coordinates and pixel references. These functions can be found through the command catalog (↱ _CAT ).

Some examples are shown below:



```
:C→PX((2.,3.))
          {# 55h,# 2h}
: PX→C({# Ah,# 102h})
           (-5.5,-22.6)
```

# Chapter 20
# Customizing menus and keyboard

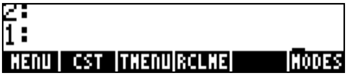
Through the use of the many calculator menus you have become familiar with the operation of menus for a variety of applications. Also, you are familiar with the many functions available by using the keys in the keyboard, whether through their main function, or by combining them with the left-shift (↰), right-shift (↱) or ALPHA (ALPHA) keys. In this Chapter we provide examples of customized menus and keyboard keys that you may find useful in your own applications.

## Customizing menus

A custom menu is a menu created by the user. The specifications for the menu are stored into the reserved variables CST. Thus, to create a menu you must put together this variable with the features that you want to display in your menu and the actions required by the soft menu keys. To show examples of customizing menus we need to set system flag 117 to SOFT menu. Make sure you do this before continuing (See Chapter 2 for instructions on setting system flags).

### The PRG/MODES/MENU menu

Commands useful in customizing menus are provided by the MENU menu, accessible through the PRG menu (↰ PRG ). Setting system flag 117 to SOFT menu, the sequence ↰ PRG NXT MODES MENU produces the following MENU soft menu:



2:
1:

The functions available are:

MENU: Activates a menu given its number

CST: Reference to the CST variable, e.g., ↱ CST shows CST contents.

TMENU: Use instead of MENU to create a temporary menu without overwriting the contents of CST

RCLMENU: Returns menu number of current menu

## Menu numbers (RCLMENU and MENU functions)

Each pre-defined menu has a number attached to it. For example, suppose that you activate the MTH menu (↰ MTH ). Then, using the function catalog (↱ CAT ) find function RCLMENU and activate it. In ALG mode simple press ENTER after `RCLMENU()` shows up in the screen. The result is the number 3.01. Thus, you can activate the MTH menu by using `MENU(3.01)`, in ALG, or `3.01 MENU`, in RPN.

Most menus can be activated without knowing their numbers by using the keyboard. There are, however, some menus not accessible through the keyboard. For example, the soft menu STATS is only accessible by using function MENU. Its number is 96.01. Use `MENU(96.01)` in ALG mode, or `96.01 MENU` in RPN mode to obtain the STAT soft menu.

**Note**: The number 96.01 in this example means the first (01) sub-menu of menu 96.

## Custom menus (MENU and TMENU functions)

Suppose that you need to activate four functions for a particular application. Say, that you need to be able to quickly access the functions EXP, LN, GAMMA and ! (ALPHA ↱ 2) and you want to place them in a soft menu that you will keep active for a while. You could do this by creating a temporary menu with function TMENU, or a more permanent menu with function MENU. The main difference is that function MENU creates variable CST, while TMENU does not. With variable CST created permanently in your sub-directory you can always reactivate the menu using the specifications in CST by pressing ↰ CUSTOM . With TMENU the menu specifications are lost after you replace the temporary menu with another one.

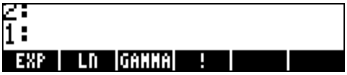
For example, in RPN mode, a menu is created by using:

{EXP LN GAMMA !} ENTER TMENU ENTER

or

{EXP LN GAMMA !} ENTER MENU ENTER

to produce the following menu:



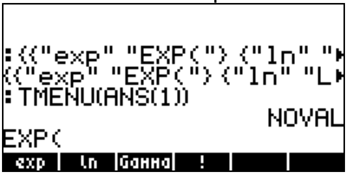
```
2:
1:
 EXP | LN |GAMMA|  !  |     |
```

To activate any of those functions you simply need to enter the function argument (a number), and then press the corresponding soft menu key.

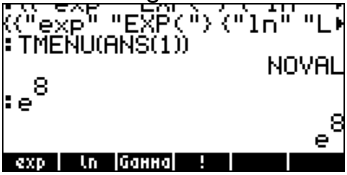
In ALG mode, the list to be entered as argument of function TMENU or MENU is more complicated:

{{“exp”,”EXP(“},{“ln”,”LN(“},{“Gamma”,”GAMMA(“},{“!”,”!(“}}

The reason for this is that, in RPN mode, the command names are both soft menu labels and commands. In ALG mode, the command names will produce no action since ALG functions must be followed by parentheses and arguments. In the list shown above (for the ALG mode), within each sub-list you have a label for the key, e.g., “exp”, followed by the way that the function will be entered in the stack so that the argument to the function can be typed at the prompt, e.g., “EXP(“. We need not worry about the closing parenthesis, because the calculator will complete the parentheses before executing the function. The implementation of function TMENU in ALG mode with the argument list shown above is as follows. First, we enter the list, then we produce the temporary menu (see menu key labels) by using function `TMENU(ANS(1))`. We also show, in the left-hand side, the result of pressing the exp soft menu key, i.e., the prompt `EXP(`. After typing 8 ENTER the result of the operation is shown in the right-hand side:



```
:{{"exp" "EXP("} {"ln" "▸
{{"exp" "EXP("} {"ln" "L▸
:TMENU(ANS(1))
                    NOVAL
EXP(
```



```
{{"exp" "EXP("} {"ln" "L▸
:TMENU(ANS(1))
                    NOVAL
:e^8
                      e^8
```

A simpler version of the menu can be defined by using

MENU({{”EXP(“,“LN(“,“GAMMA(“,”!(“}).

**Enhanced RPN menu**

The list presented above for the ALG mode, can be modified slightly to use in the RPN mode. The modified list will look like this:

{{“exp”,EXP},{“ln”,LN},{“Gamma”,GAMMA},{“!”,!}}

You can try using this list with TMENU or MENU in RPN mode to verify that you get the same menu as obtained earlier in ALG mode.

## Menu specification and CST variable

From the two exercises shown above we notice that the most general menu specification list include a number of sub-lists equal to the number of items to be displayed in your custom menu. Each sub-list contains a label for the menu key followed by a function, expression, label, or other object that constitutes the effect of the menu key when pressed. Care must be exercised in specifying the menu list in ALG mode versus RPN mode. In RPN mode, the menu key action can be simply a calculator command (e.g., EXP, LN, etc., as shown above), while in ALG mode it has to be a string with the command prompt whose argument needs to be provided by the user before pressing ENTER and completing the command. The examples above illustrate the difference.

The general form of the argument list for commands TMENU or MENU in ALG mode is
{“label1”,”function1(“,”ls1(“,”rs1(“}, {“label2”, “function2(“,”ls2(“,”rs2(“},…}

While, in RPN mode, the argument list has this format
{“label1”, function1, ls1, rs1}, {“label2”, function2, ls2, rs2},…}

In these specifications, function1, function 2, etc., represent the main operation of the key, while ls1, ls2, …, etc., represent the left-shift operation of the key. Similarly, rs1, rs2, …, etc., represent the right-shift operation of the key. This list will be stored in variable CST if command MENU is used. You can have a different CST variable in each sub-directory, and you can always replace the current contents of CST with those of other variables storing the properly formatted list to produce another custom menu.

**Note**: You can use a 21x8 GROB (See Chapter 22) to produce an icon in the soft menu keys. As an example, try, in RPN mode:

{{GROB 21 8 00000EF908FFF900FFF9B3FFF9A2FFF9A3FFF9A0FFF388FF “hp” }}
ENTER MENU

This will place the hp logo on key F1 . Pressing F1 places the text ‘hp’ in the command line.

# Customizing the keyboard

Each key in the keyboard can be identified by two numbers representing their row and column. For example, the VAR key (VAR) is located in row 3 of column 1, and will be referred to as key 31. Now, since each key has up to ten functions associated with it, each function is specified by decimal digits between 0 and 1, according to the following specifications:

.0 or 1, unshifted key
.2, key combined with ↰
.3, key combined with ↱
.4, key combined with ALPHA
.5, key combined with ALPHA ↰
.6, key combined with ALPHA ↱

0.01 or 0.11, not applicable
.21, key simultaneous with ↰
.31, key simultaneous with ↱
.41, key combined with ALPHA
.51, ALPHA key simultaneous with ↰
.61, ALPHA key simultaneous with ↱

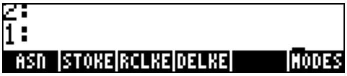
Thus, the VAR function will be referred to as key 31.0 or 31.1, while the UPDIR function will be key 31.2, the COPY function will be key 31.3, the upper-case J is key 31.4, and lower case j is key 31.5. (Key 31.6 is not defined). In general, a key will be described by the arrangement XY.Z, where X = row number, Y = column number, Z = shifting.

We can combine a given key with the USER key (left-shift associated with the ALPHA key, or ↰ USER ) to create a customized key action. In principle, the entire keyboard can be re-defined to perform a number of customized operations.

## The PRG/MODES/KEYS sub-menu

Commands useful in customizing the keyboard are provided by the KEYS menu accessible through the PRG menu (↰ PRG ). Setting system flag 117 to SOFT menu, the sequence ↰ PRG NXT MODES KEYS

produces the following KEYS soft menu:



```
2:
1:
 ASN  STOKE RCLKE DELKE       MODES
```

The functions available are:

ASN: Assigns an object to a key specified by XY.Z
STOKEYS: Stores user-defined key list
RCLKEYS: Returns current user-defined key list
DELKEYS: Un-assigns one or more keys in the current user-defined key list, the arguments are either 0, to un-assign all user-defined keys, or XY.Z, to un-assign key XY.Z.

## Recall current user-defined key list

Use command RCLKEYS to see the current user-defined key list. Before any user-defined key assignments, the result should be a list containing the letter S, i.e., {S}.

## Assign an object to a user-defined key

Suppose that you want to have access to the old-fashioned PLOT command first introduced with the HP 48G series calculator, but currently not directly available from the keyboard. The menu number for this menu is 81.01. You can see this menu active by using

ALG mode: MENU(81.01)
RPN mode: 81.01 ENTER MENU ENTER

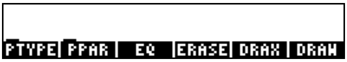
If you want to have a quick way to activate this menu from the keyboard, you could assign this menu to the GRAPH key ( F3 ) whose reference number is 13.0, i.e., first row, third column, main function. To assign an object to a key use function ASN, as follows:

ALG mode: ASN(<<MENU(81.01)>>,13.0)
RPN mode: << 18.01 MENU >> ENTER 13.0 ENTER ASN

Another useful menu is the original SOLVE menu (described at the end of Chapter 6 in this Guide), which can be activated by using ↱ (hold) 7 .

## Operating user-defined keys

To operate this user-defined key, enter ↰ USER before pressing the F3 key. Notice that after pressing ↰ USER the screen shows the specification 1USR in the second display line. Pressing for ←USER F3 for this example, you should recover the PLOT menu as follows:



PTYPE PPAR EQ ERASE DRAX DRAW

If you have more than one user-defined key and want to operate more than one of them at a time, you can lock the keyboard in USER mode by entering ←USER ←USER before pressing the user-defined keys. With the keyboard locked in USER mode, the specification USR will be shown in the second display line. To unlock the keyboard press ←USER once more.

## Un-assigning a user-defined key

To remove the assignment performed above, use function DELKEYS, as follows:

ALG mode: `DELKEYS(13.0)`

RPN mode: `13.0` ENTER `DELKEYS` ENTER

## Assigning multiple user-defined keys

The simplest way to assign several user-defined is to provide a list of commands and key specifications. For example, suppose that we assign the three trigonometric functions (SIN, COS, TAN) and the three hyperbolic functions (SINH, COSH, TANH) to keys F1 through F6, respectively, as user-defined keys. In RPN mode use:

`{SIN,11.0,COS,12.0,TAN,13.0,SINH,14.0,COSH,15.0,TANH,16.0}` ENTER `STOKEYS` ENTER

In ALG mode use:

`STOKEYS({"SIN(' ,11.0, "COS(", 12.0, "TAN(", 13.0, "SINH(", 14.0, "COSH(", 15.0, "TANH(", 16.0})` ENTER

Operate these keys by using, for example, in RPN mode:

5 ←USER F1    4 ←USER F2    6 ←USER F3
2 ←USER F4    1 ←USER F5    2 ←USER F6

To un-assign all user-defined keys use:

ALG mode: `DELKEYS(0)`    RPN mode: `0 DELKEYS`

Check that the user-key definitions were removed by using function RCLKEYS.

# Chapter 21
# Programming in User RPL language

User RPL language is the programming language most commonly used to program the calculator. The program components can be put together in the line editor by including them between program containers « » in the appropriate order. Because there is more experience among calculator users in programming in the RPN mode, most of the examples in this Chapter will be presented in the RPN mode. Also, to facilitate entering programming commands, we suggest you set system flag 117 to SOFT menus. The programs work equally well in ALG mode once they have been debugged and tested in RPN mode. If you prefer to work in the ALG mode, simply learn how to do the programming in RPN and then reset the operating mode to ALG to run the programs. For a simple example of User RPL programming in ALG mode, refer to the last page in this chapter.

## An example of programming

Throughout the previous Chapters in this guide we have presented a number of programs that can be used for a variety of applications (e.g., programs CRMC and CRMT, used to create a matrix out of a number of lists, were presented in Chapter 10). In this section we present a simple program to introduce concepts related to programming the calculator. The program we will write will be used to define the function *f(x) = sinh(x)/(1+x²)*, which accepts lists as argument (i.e., x can be a list of numbers, as described in Chapter 8). In Chapter 8 we indicated that the plus sign, , acts as a concatenation operator for lists and not to produce a term-by-term sum. Instead, you need to use the ADD operator to achieve a term-by-term summation of lists. Thus, to define the function shown above we will use the following program:

```
        «'x' STO x SINH 1 x SQ ADD / 'x' PURGE »
```

To key in the program follow these instructions:

| Keystroke sequence: | Produces: | Interpreted as: |
|---|---|---|
| ↱ «» | « | Start an RPL program |
| ['] ALPHA ↰ X ▶ STO▶ | 'x' STO | Store level 1 into variable x |
| ALPHA ↰ X | x | Place x in level 1 |

| | | |
|---|---|---|
| ↰ MTH HYP SINH | SINH | Calculate sinh of level 1 |
| 1 SPC ALPHA ↰ X ↰ x² | 1 x SQ | Enter 1 and calculate $x^2$ |
| ↰ MTH MTH LIST ADD | ADD | Calculate $(1+x^2)$, |
| ÷ | / | then divide |
| ['] ALPHA ↰ X ▶ | 'x' | |
| ↰ PRG MEM DIR PURGE | PURGE | Purge variable x |
| ENTER | | Program in level 1 |

To save the program use: ['] ALPHA ↰ G STO▶

Press VAR to recover your variable menu, and evaluate g(3.5) by entering the value of the argument in level 1 (3 . 5 ENTER) and then pressing g. The result is 1.2485…, i.e., g(3.5) = 1.2485. Try also obtaining g({1 2 3}), by entering the list in level 1 of the display: ↰ {} 1 SPC 2 SPC 3 ENTER and pressing g. The result now is {SINH(1)/2 SINH(2)/5 SINH(3)/10}, if your CAS is set to EXACT mode. If your CAS is set to APPROXIMATE mode, the result will be {0.5876.. 0.7253… 1.0017…}.

## Global and local variables and subprograms

The program g, defined above, can be displayed as

```
« 'x' STO x SINH 1 x SQ ADD / 'x' PURGE »
```

by using ↱ g.

Notice that the program uses the variable name x to store the value placed in level 1 of stack through the programming steps 'x' STO. The variable x, while the program is executing, is stored in your variable menu as any other variable you had previously stored. After calculating the function, the program purges (erases) the variable x so it will not show in your variable menu after finishing evaluating the program. If we were not to purge the variable x within the program its value would be available to us after program execution. For that reason, the variable x, as used in this program, is referred to as *a global variable*. One implication of the use of x as a global variable is that, if we had a previously defined a variable with the name x, its value would be replaced by the value that the program uses and then completely removed from your variable menu after program execution.

From the point of view of programming, therefore, a _global variable_ is a variable that is accessible to the user after program execution. It is possible to use a local variable within the program that is only defined for that program and will not be available for use after program execution. The previous program could be modified to read:

```
« → x « x SINH 1 x SQ ADD / »  »
```

The arrow symbol (→) is obtained by combining the right-shift key ↱ with the 0 key, i.e., ↱ → . Also, notice that there is an additional set of programming symbols (« ») indicating the existence of a _sub-program_, namely « `x SINH 1 x SQ ADD /` », within the main program. The main program starts with the combination → `x`, which represents assigning the value in level 1 of stack to a _local variable_ x. Then, programming flow continues within the sub-program by placing *x* in the stack, evaluating *SINH(x),* placing *1* in the stack, placing *x* in the stack, squaring *x*, adding *1* to *x*, and dividing stack level 2 (*SINH(x)*) by stack level 1 ($1+x^2$). The program control is then passed back to the main program, but there are no more commands between the first set of closing programming symbols (») and the second one, therefore, the program terminates. The last value in the stack, i.e., *SINH(x)/* ($1+x^2$), is returned as the program output.

The variable x in the last version of the program never occupies a place among the variables in your variable menu. It is operated upon within the calculator memory without affecting any similarly named variable in your variable menu. For that reason, the variable x in this case is referred to as a variable local to the program, i.e., a _local variable_.

**Note**: To modify program g, place the program name in the stack (' g ENTER), then use ↰ ▼. Use the arrow keys (◀ ▶ ▲ ▼) to move about the program. Use the backspace/delete key, ⬅, to delete any unwanted characters. To add program containers (i.e., « »), use ↱ « », since these symbols come in pairs you will have to enter them at the start and end of the sub-program and delete one of its components with the delete key ⬅ to produce the required program, namely:

```
« → x « x SINH 1 x SQ ADD / » ».
```

When done editing the program press ENTER . The modified program is stored back into variable [illegible].

## Global Variable Scope

Any variable that you define in the HOME directory or any other directory or sub-directory will be considered a *global variable* from the point of view of program development. However, the *scope* of such variable, i.e., *the location in the directory tree where the variable is accessible*, will depend on the location of the variable within the tree (see Chapter 2).

The rule to determine a variable's scope is the following*: a global variable is accessible to the directory where it is defined and to any sub-directory attached to that directory, unless a variable with the same name exists in the sub-directory under consideration.* Consequences of this rule are the following:

- A global variable defined in the HOME directory will be accessible from any directory within HOME, unless redefined within a directory or sub-directory.
- If you re-define the variable within a directory or sub-directory this definition takes precedence over any other definition in directories above the current one.
- When running a program that references a given global variable, the program will use the value of the global variable in the directory from which the program is invoked. If no variable with that name exist in the invoking directory, the program will search the directories above the current one, up to the HOME directory, and use the value corresponding to the variable name under consideration in the closest directory above the current one.

A program defined in a given directory can be accessed from that directory or any of its sub-directories.

All these rule may sound confusing for a new calculator user. They all can be simplified to the following suggestion: *Create directories and sub-directories with meaningful names to organize your data, and make sure you have all the global variables you need within the proper sub-directory.*

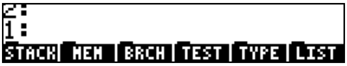
## Local Variable Scope

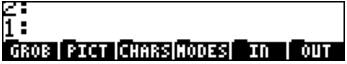
Local variables are active only within a program or sub-program. Therefore, their scope is limited to the program or sub-program where they're defined. An example of a local variable is the index in a FOR loop (described later in this chapter), for example `« → n x « 1 n FOR j x NEXT n →LIST » »`

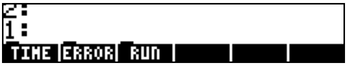
# The PRG menu

In this section we present the contents of the PRG (programming) menu with the calculator's system flag 117 set to SOFT menus. With this flag setting sub-menus and commands in the PRG menu will be shown as soft menu labels. This facilitates entering the programming commands in the line editor when you are putting together a program.

To access the PRG menu use the keystroke combination ← PRG . Within the PRG menu we identify the following sub-menus (press NXT to move to the next collection of sub-menus in the PRG menu):







Here is a brief description of the contents of these sub-menus, and their sub-menus:

STACK: Functions for manipulating elements of the RPN stack
MEM: Functions related to memory manipulation
  DIR: Functions related to manipulating directories
  ARITH: Functions to manipulate indices stored in variables
BRCH: Collection of sub-menus with program branching and loop functions
  IF: IF-THEN-ELSE-END construct for branching
  CASE: CASE-THEN-END construct for branching
  START: START-NEXT-STEP construct for branching
  FOR: FOR-NEXT-STEP construct for loops

DO: DO-UNTIL-END construct for loops
WHILE: WHILE-REPEAT-END construct for loops
TEST: Comparison operators, logical operators, flag testing functions
TYPE: Functions for converting object types, splitting objects, etc.
LIST: Functions related to list manipulation
ELEM: Functions for manipulating elements of a list
PROC: Functions for applying procedures to lists
GROB: Functions for the manipulation of graphic objects
PICT: Functions for drawing pictures in the graphics screen
CHARS:Functions for character string manipulation
MODES: Functions for modifying calculator modes
FMT: To change number formats, comma format
ANGLE:To change angle measure and coordinate systems
FLAG: To set and un-set flags and check their status
KEYS: To define and activate user-defined keys (Chapter 20)
MENU: To define and activate custom menus (Chapter 20)
MISC: Miscellaneous mode changes (beep, clock, etc.)
IN: Functions for program input
OUT: Functions for program output
TIME: Time-related functions
ALRM: Alarm manipulation
ERROR: Functions for error handling
IFERR: IFERR-THEN-ELSE-END construct for error handling
RUN: Functions for running and debugging programs

## Navigating through RPN sub-menus

Start with the keystroke combination ← PRG , then press the appropriate soft-menu key (e.g., MEM ). If you want to access a sub-menu within this sub-menu (e.g., DIR within the MEM sub-menu), press the corresponding key. To move up in a sub-menu, press the NXT key until you find either the reference to the upper sub-menu (e.g., MEM within the DIR sub-menu) or to the PRG menu (i.e., PRG ).

## Functions listed by sub-menu

The following is a listing of the functions within the PRG sub-menus listed by sub-menu.

**STACK**

- DUP
- SWAP
- DROP
- OVER
- ROT
- UNROT
- ROLL
- ROLLD
- PICK
- UNPICK
- PICK3
- DEPTH
- DUP2
- DUPN
- DROP2
- DROPN
- DUPDU
- NIP
- NDUPN

**MEM**

- PURGE
- MEM
- BYTES
- NEWOB
- ARCHI
- RESTO

**MEM/DIR**

- PURGE
- RCL
- STO
- PATH
- CRDIR
- PGDIR
- VARS
- TVARS
- ORDER

**MEM/ARITH**

- STO+
- STO-
- STOx
- STO/
- INCR
- DECR
- SINV
- SNEG
- SCONJ

**BRCH**

- IFT
- IFTE

**BRCH/IF**

- IF
- THEN
- ELSE
- END

**BRCH/CASE**

- CASE
- THEN
- END

**BRCH/START**

- START
- NEXT
- STEP

**BRCH/FOR**

- FOR
- NEXT
- STEP

**BRCH/DO**

- DO
- UNTIL
- END

**BRCH/WHILE**

- WHILE
- REPEAT
- END

**TEST**

- ==
- ≠
- <
- >
- ≤
- ≥
- AND
- OR
- XOR
- NOT
- SAME
- TYPE
- SF
- CF
- FS?
- FC?
- FS?C
- FC?C
- LININ

**TYPE**

- OBJ→
- →ARRY
- →LIST
- →STR
- →TAG
- →UNIT
- C→R
- R→C
- NUM
- CHR
- DTAG
- EQ→
- TYPE
- VTYPE

**LIST**

- OBJ→
- →LIST
- SUB
- REPL

**LIST/ELEM**
GET
GETI
PUT
PUTI
SIZE
POS
HEAD
TAIL

**LIST/PROC**
DOLIST
DOSUB
NSUB
ENDSUB
STREAM
REVLIST
SORT
SEQ

**GROB**
→GROB
BLANK
GOR
GXOR
SUB
REPL
→LCD
LCD→
SIZE
ANIMATE

**PICT**
PICT
PDIM
LINE
TLINE
BOX
ARC
PIXON
PIXOF
PIX?
PVIEW
PX→C
C→PX

**CHARS**
SUB
REPL
POS
SIZE
NUM
CHR
OBJ→
→STR
HEAD
TAIL
SREPL

**MODES/FMT**
STD
FIX
SCI
ENG
FM,
ML

**MODES/ANGLE**
DEG
RAD
GRAD
RECT
CYLIN
SPHERE

**MODES/FLAG**
SF
CF
FS?
FC?
FS?C
FS?C
FC?C
STOF
RCLF
RESET

**MODES/KEYS**
ASN
STOKEYS
RECLKEYS
DELKEYS

**MODES/MENU**
MENU
CST
TMENU
RCLMENU

**MODES/MISC**
BEEP
CLK
SYM
STK
ARG
CMD
INFO

**IN**
INFORM
NOVAL
CHOOSE
INPUT
KEY
WAIT
PROMPT

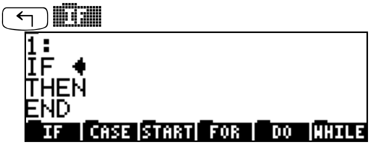
**OUT**
PVIEW
TEXT
CLLCD
DISP
FREEZE
MSGBOX
BEEP

| **TIME** | **ERROR** | **RUN** |
|---|---|---|
| DATE | DOERR | DBUG |
| →DATE | ERRN | SST |
| TIME | ERRM | SST↓ |
| →TIME | ERR0 | NEXT |
| TICKS | LASTARG | HALT |
| | | KILL |
| **TIME/ALRM** | **ERROR/IFERR** | OFF |
| ACK | IFERR | |
| ACKALARM | THEN | |
| STOALARM | ELSE | |
| RCLALARM | END | |
| DELALARM | | |
| FINDALARM | | |

## Shortcuts in the PRG menu

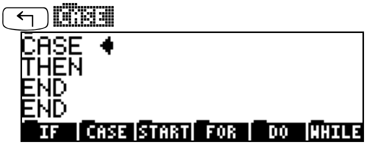
Many of the functions listed above for the PRG menu are readily available through other means:

- Comparison operators (≠, ≤, <, ≥, >) are available in the keyboard.
- Many functions and settings in the MODES sub-menu can be activated by using the input functions provided by the MODE key.
- Functions from the TIME sub-menu can be accessed through the keystroke combination ↱ TIME .
- Functions STO and RCL (in MEM/DIR sub-menu) are available in the keyboard through the keys STO▸ and ↰ RCL .
- Functions RCL and PURGE (in MEM/DIR sub-menu) are available through the TOOL menu (TOOL).
- Within the BRCH sub-menu, pressing the left-shift key (↰) or the right-shift key (↱) before pressing any of the sub-menu keys, will create constructs related to the sub-menu key chosen. This only works with the calculator in RPN mode. Examples are shown below:



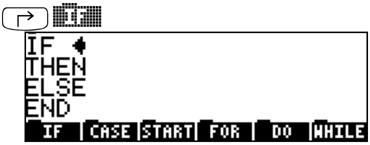
↰ IF

```
1:
IF ◀
THEN
END
 IF  CASE START FOR  DO  WHILE
```



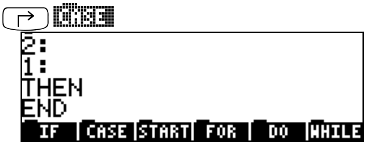
↰ CASE

```
CASE ◀
THEN
END
END
 IF  CASE START FOR  DO  WHILE
```



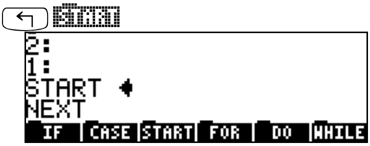
↱ IF

```
IF ◀
THEN
ELSE
END
 IF  CASE START FOR  DO  WHILE
```



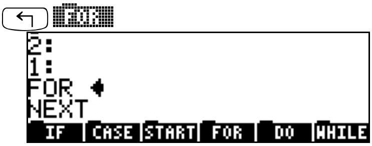
↱ CASE

```
2:
1:
THEN
END
 IF  CASE START FOR  DO  WHILE
```



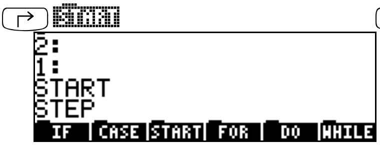
↰ START

```
2:
1:
START ◀
NEXT
 IF  CASE START FOR  DO  WHILE
```



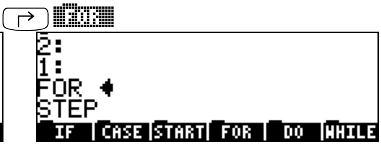
↰ FOR

```
2:
1:
FOR ◀
NEXT
 IF  CASE START FOR  DO  WHILE
```



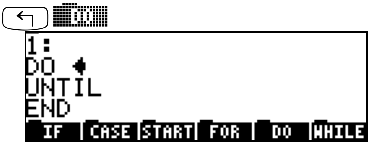
↱ START

```
2:
1:
START
STEP
 IF  CASE START FOR  DO  WHILE
```



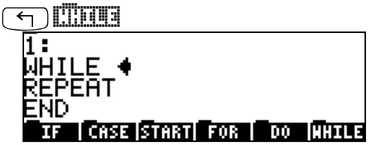
↱ FOR

```
2:
1:
FOR ◀
STEP
 IF  CASE START FOR  DO  WHILE
```



↰ DO

```
1:
DO ◀
UNTIL
END
 IF  CASE START FOR  DO  WHILE
```



↰ WHILE

```
1:
WHILE ◀
REPEAT
END
 IF  CASE START FOR  DO  WHILE
```

Notice that the insert prompt (◀) is available after the key word for each construct so you can start typing at the right location.

## Keystroke sequence for commonly used commands

The following are keystroke sequences to access commonly used commands for numerical programming within the PRG menu. The commands are first listed by menu:

| | | |
|---|---|---|
| STACK | | |
| | DUP | ← PRG STACK DUP |
| | SWAP | ← PRG STACK SWAP |
| | DROP | ← PRG STACK DROP |
| MEM DIR | | |
| | PURGE | ← PRG MEM DIR PURGE |
| | ORDER | ← PRG MEM DIR ORDER |
| BRCH IF | | |
| | IF | ← PRG BRCH IF IF |
| | THEN | ← PRG BRCH IF THEN |
| | ELSE | ← PRG BRCH IF ELSE |
| | END | ← PRG BRCH IF END |
| BRCH CASE | | |
| | CASE | ← PRG BRCH CASE CASE |
| | THEN | ← PRG BRCH CASE THEN |
| | END | ← PRG BRCH CASE END |
| BRCH START | | |
| | START | ← PRG BRCH START START |
| | NEXT | ← PRG BRCH START NEXT |
| | STEP | ← PRG BRCH START STEP |
| BRCH FOR | | |
| | FOR | ← PRG BRCH FOR FOR |
| | NEXT | ← PRG BRCH FOR NEXT |
| | STEP | ← PRG BRCH FOR STEP |
| BRCH DO | | |
| | DO | ← PRG BRCH DO DO |
| | UNTIL | ← PRG BRCH DO UNTIL |
| | END | ← PRG BRCH DO END |

BRCH WHILE

| | |
|---|---|
| WHILE | ↰ PRG BRCH WHILE WHILE |
| REPEAT | ↰ PRG BRCH WHILE REPEA |
| END | ↰ PRG BRCH WHILE END |

TEST

| | |
|---|---|
| == | ↰ PRG TEST ≠ |
| AND | ↰ PRG TEST NXT AND |
| OR | ↰ PRG TEST NXT OR |
| XOR | ↰ PRG TEST NXT XOR |
| NOT | ↰ PRG TEST NXT NOT |
| SAME | ↰ PRG TEST NXT SAME |
| SF | ↰ PRG TEST NXT NXT SF |
| CF | ↰ PRG TEST NXT NXT CF |
| FS? | ↰ PRG TEST NXT NXT FS? |
| FC? | ↰ PRG TEST NXT NXT FC? |
| FS?C | ↰ PRG TEST NXT NXT FS?C |
| FC?C | ↰ PRG TEST NXT NXT FC?C |

TYPE

| | |
|---|---|
| OBJ→ | ↰ PRG TYPE OBJ→ |
| →ARRY | ↰ PRG TYPE →ARRY |
| →LIST | ↰ PRG TYPE →LIST |
| →STR | ↰ PRG TYPE →STR |
| →TAG | ↰ PRG TYPE →TAG |
| NUM | ↰ PRG TYPE NXT NUM |
| CHR | ↰ PRG TYPE NXT CHR |
| TYPE | ↰ PRG TYPE NXT TYPE |

LIST ELEM

| | |
|---|---|
| GET | ↰ PRG LIST ELEM GET |
| GETI | ↰ PRG LIST ELEM GETI |
| PUT | ↰ PRG LIST ELEM PUT |
| PUTI | ↰ PRG LIST ELEM PUTI |
| SIZE | ↰ PRG LIST ELEM SIZE |
| HEAD | ↰ PRG LIST ELEM NXT HEAD |
| TAIL | ↰ PRG LIST ELEM NXT TAIL |

| | |
|---|---|
| LIST PROC | |
| REVLIST | ← PRG LIST PROC REVLI |
| SORT | ← PRG LIST PROC NXT SORT |
| SEQ | ← PRG LIST PROC NXT SEQ |
| MODES ANGL | |
| DEG | ← PRG NXT MODES ANGL DEG |
| RAD | ← PRG NXT MODES ANGL DEG |
| MODES MENU | |
| CST | ← PRG NXT MODES MENU CST |
| MENU | ← PRG NXT MODES MENU MENU |
| BEEP | ← PRG NXT MODES MISC BEEP |
| IN | |
| INFORM | ← PRG NXT IN INFOR |
| INPUT | ← PRG NXT IN INPUT |
| MSGBOX | ← PRG NXT OUT MSGBO |
| PVIEW | ← PRG NXT OUT PVIEW |
| RUN | |
| DBUG | ← PRG NXT NXT RUN DBG |
| SST | ← PRG NXT NXT RUN SST |
| SST↓ | ← PRG NXT NXT RUN SST↓ |
| HALT | ← PRG NXT NXT RUN HALT |
| KILL | ← PRG NXT NXT RUN KILL |

## Programs for generating lists of numbers

Please notice that the functions in the PRG menu are not the only functions that can be used in programming. As a matter of fact, almost all functions in the calculator can be included in a program. Thus, you can use, for example, functions from the MTH menu. Specifically, you can use functions for list operations such as SORT, ΣLIST, etc., available through the MTH/LIST menu.

As additional programming exercises, and to try the keystroke sequences listed above, we present herein three programs for creating or manipulating lists. The program names and listings are as follows:

***LISC***:

```
« → n x « 1 n FOR j x NEXT n →LIST » »
```

***CRLST***:

```
« → st en df « st en FOR  j j df STEP en st - df / FLOOR 1 +
→LIST » »
```

***CLIST***:

```
« REVLIST DUP DUP SIZE 'n' STO ΣLIST SWAP TAIL DUP SIZE 1 - 1
SWAP FOR j DUP ΣLIST SWAP TAIL NEXT 1 GET n →LIST REVLIST 'n'
PURGE »
```

The operation of these programs is as follows:

(1) *LISC*: creates a list of n elements all equals to a constant c.
*Operation*: enter n, enter c, press LISC
*Example*: 5 ENTER 6.5 ENTER LISC creates the list: {6.5 6.5 6.5 6.5 6.5}

(2) *CRLST*: creates a list of numbers from $n_1$ to $n_2$ with increment $\Delta n$, i.e., $\{n_1, n_1+\Delta n, n1+2\cdot\Delta n, \ldots n_1+N\cdot\Delta n\}$, where $N=\text{floor}((n_2-n_1)/\Delta n)+1$.
*Operation*: enter $n_1$, enter $n_2$, enter $\Delta n$, press CRLST
*Example*: .5 ENTER 3.5 ENTER .5 ENTER CRLST produces: {0.5 1 1.5 2 2.5 3 3.5}

(3) *CLIST*: creates a list with cumulative sums of the elements, i.e., if the original list is $\{x_1\ x_2\ x_3\ \ldots\ x_N\}$, then CLIST creates the list:

$$\{x_1, x_1 + x_2, x_1 + x_2 + x_3, \ldots, \sum_{i=1}^{N} x_i\}$$

*Operation*: place the original list in level 1, press CLIST.
*Example*: {1 2 3 4 5} ENTER CLIST produces {1 3 6 10 15}.

# Examples of sequential programming

In general, a program is any sequence of calculator instructions enclosed between the program containers « and ». Subprograms can be included as part of a program. The examples presented previously in this guide (e.g., in Chapters 3 and 8) 6 can be classified basically into two types: (a) programs generated by defining a function; and, (b) programs that simulate a sequence of stack operations. These two types of programs are described next. The general form of these programs is input→process→output, therefore, we refer to them as <u>sequential programs</u>.

## Programs generated by defining a function

These are programs generated by using function DEFINE (↰ DEF ) with an argument of the form:

'function_name($x_1$, $x_2$, …) = expression containing variables $x_1$, $x_2$, …'

The program is stored in a variable called `function_name`. When the program is recalled to the stack, by using ↱ function_name. The program shows up as follows:

« → $x_1$, $x_2$, … 'expression containing variables $x_1$, $x_2$, …'».

To evaluate the function for a set of input variables $x_1$, $x_2$, …, in RPN mode, enter the variables into the stack in the appropriate order (i.e., $x_1$ first, followed by $x_2$, then $x_3$, etc.), and press the soft menu key labeled function_name. The calculator will return the value of the function function_name($x_1$, $x_2$, …).

<u>*Example*</u>: *Manning's equation for wide rectangular channel.*

As an example, consider the following equation that calculates the unit discharge (discharge per unit width), q, in a wide rectangular open channel using Manning's equation:

$$q = \frac{C_u}{n} y_0^{5/3} \sqrt{S_0}$$

where $C_u$ is a constant that depends on the system of units used [$C_u$ = 1.0 for units of the International System (S.I.), and $C_u$ = 1.486 for units of the English System (E.S.)], n is the Manning's resistance coefficient, which depends on the type of channel lining and other factors, $y_0$ is the flow depth, and $S_0$ is the channel bed slope given as a dimensionless fraction.

**Note**: Values of the Manning's coefficient, n, are available in tables as dimensionless numbers, typically between 0.001 to 0.5. The value of Cu is also used without dimensions. However, care should be taken to ensure that the value of y0 has the proper units, i.e., m in S.I. and ft in E.S. The result for q is returned in the proper units of the corresponding system in use, i.e., $m^2/s$ in S.I. and $ft^2/s$ in E.S. Manning's equation is, therefore, not *dimensionally consistent.*

Suppose that we want to create a function q(Cu, n, y0, S0) to calculate the unit discharge q for this case. Use the expression

'q(Cu,n,y0,S0)=Cu/n*y0^(5./3.)*√S0',

as the argument of function DEFINE. Notice that the exponent 5./3., in the equation, represents a ratio of real numbers due to the decimal points. Press VAR, if needed, to recover the variable list. At this point there will be a variable called q in your soft menu key labels. To see the contents of q, use ↱ q. The program generated by defining the function `q(Cu,n,y0,S0)` is shown as:

« → Cu n y0 S0 'Cu/n*y0^(5./3.)*√S0' ».

This is to be interpreted as "enter Cu, n, y0, S0, in that order, then calculate the expression between quotes." For example, to calculate q for Cu = 1.0, n = 0.012, y0 = 2 m, and S0 = 0.0001, use, in RPN mode:

1 ENTER 0.012 ENTER 2 ENTER 0.0001 ENTER q

The result is 2.6456684 (or, q = 2.6456684 $m^2/s$).

You can also separate the input data with spaces in a single stack line rather than using ENTER .

## Programs that simulate a sequence of stack operations

In this case, the terms to be involved in the sequence of operations are assumed to be present in the stack. The program is typed in by first opening the program containers with ↱ « » . Next, the sequence of operations to be performed is entered. When all the operations have been typed in, press ENTER to complete the program. If this is to be a once-only program, you can at this point, press EVAL to execute the program using the input data available. If it is to be a permanent program, it needs to be stored in a variable name.

The best way to describe this type of programs is with an example:

_Example_: *Velocity head for a rectangular channel.*
Suppose that we want to calculate the velocity head, $h_v$, in a rectangular channel of width b, with a flow depth y, that carries a discharge Q. The specific energy is calculated as $h_v = Q^2/(2g(by)^2)$, where g is the acceleration of gravity ($g = 9.806\ m/s^2$ in S.I. units or $g = 32.2\ ft/s^2$ in E.S. units). If we were to calculate $h_v$ for Q = 23 cfs (cubic feet per second = $ft^3/s$), b = 3 ft, and y = 2 ft, we would use: $h_v = 23^2/(2\cdot 32.2\cdot\ (3\cdot 2)^2)$. Using the RPN modethe calculator, interactively, we can calculate this quantity as:

2 ENTER 3 × ↰ $x^2$ 3 2 • 2 ×
2 × 2 3 ↰ $x^2$ ▶ ÷

Resulting in 0.228174, or $h_v = 0.228174$.

To put this calculation together as a program we need to have the input data (Q, g, b, y) in the stack in the order in which they will be used in the calculation. In terms of the variables Q, g, b, and y, the calculation just performed is written as (do not type the following):

y ENTER b × ↰ $x^2$ g × 2 × Q ↰ $x^2$ ▶ ÷

As you can see, y is used first, then we use b, g, and Q, in that order. Therefore, for the purpose of this calculation we need to enter the variables in the inverse order, i.e., (do not type the following):

Q ENTER g ENTER b ENTER y ENTER

For the specific values under consideration we use:

23 ENTER 32.2 ENTER 3 ENTER 2 ENTER

The program itself will contain only those keystrokes (or commands) that result from removing the input values from the interactive calculation shown earlier, i.e., removing Q, g, b, and y from (do not type the following):

y ENTER b × ↰ x² g × 2 × Q ↰ x² ▶ ÷

and keeping only the operations shown below (do not type the following):

ENTER × ↰ × 2 × ↰ x² ▶ ÷

**Note**: When entering the program do not use the keystroke ▶, instead use the keystroke sequence: ↰ PRG STACK SWAP.

Unlike the interactive use of the calculator performed earlier, we need to do some swapping of stack levels 1 and 2 within the program. To write the program, we use, therefore:

| | |
|---|---|
| ↱ « » | Opens program symbols |
| × | Multiply y with b |
| ↰ x² | Square (b·y) |
| × | Multiply (b·y)$^2$ times g |
| 2 × | Enter a 2 and multiply it with g· (b·y)$^2$ |
| ↰ PRG STACK SWAP | Swap Q with 2·g· (b·y)$^2$ |
| ↰ x² | Square Q |
| ↰ PRG STACK SWAP | Swap 2·g· (b·y)$^2$ with Q$^2$ |
| ÷ | Divide Q$^2$ by 2·g· (b·y)$^2$ |
| ENTER | Enter the program |

The resulting program looks like this:

```
« * SQ * 2 * SWAP SQ SWAP / »
```

**Note**: SQ is the function that results from the keystroke sequence ← x² .

Save the program into a variable called hv:

' ALPHA ← H ALPHA ← V STO▶

A new variable hv should be available in your soft key menu. (Press VAR to see your variable list.) The program left in the stack can be evaluated by using function EVAL. The result should be 0.228174…, as before. Also, the program is available for future use in variable hv. For example, for Q = 0.5 m$^3$/s, g = 9.806 m/s$^2$, b = 1.5 m, and y = 0.5 m, use:

0.5 SPC 1.5 SPC 9.806 SPC 1.5 SPC 0.5 hv

**Note**: SPC is used here as an alternative to ENTER for input data entry.

The result now is 2.26618623518E-2, i.e., hv = 2.26618623518×10$^{-2}$ m.

**Note:** Since the equation programmed in hv is dimensionally consistent, we can use units in the input.

As mentioned earlier, the two types of programs presented in this section are *sequential programs*, in the sense that the program flow follows a single path, i.e., INPUT→ OPERATION →OUTPUT. Branching of the program flow is possible by using the commands in the menu ← PRG BRCH . More detail on program branching is presented below.

# Interactive input in programs

In the sequential program examples shown in the previous section it is not always clear to the user the order in which the variables must be placed in the stack before program execution. For the case of the program q, written as

```
« →Cu n y0 S0 'Cu/n*y0^(5/3)*√S0' »,
```

it is always possible to recall the program definition into the stack (→ q) to see the order in which the variables must be entered, namely, → `Cu n y0 S0`. However, for the case of the program hv, its definition

« * SQ * 2 * SWAP SQ SWAP / »

does not provide a clue of the order in which the data must be entered, unless, of course, you are extremely experienced with RPN and the User RPL language.

One way to check the result of the program as a formula is to enter symbolic variables, instead of numeric results, in the stack, and let the program operate on those variables. For this approach to be effective the calculator's CAS (Calculator Algebraic System) must be set to `symbolic` and `exact` modes. This is accomplished by using MODE CAS, and ensuring that the check marks in the options `_Numeric` and `_Approx` are removed. Press OK OK to return to normal calculator display. Press VAR to display your variables menu.

We will use this latter approach to check what formula results from using the program hv as follows: We know that there are four inputs to the program, thus, we use the symbolic variables S4, S3, S2, and S1 to indicate the stack levels at input:

ALPHA S 4 ENTER ALPHA S 3 ENTER ALPHA S 2 ENTER ALPHA S 1 ENTER

Next, press hv. The resulting formula may look like this

`'SQ(S4)/(S3*SQ(S2*S1)*2)',`

if your display is not set to textbook style, or like this,

$$\frac{SQ(S4)}{S3 \cdot SQ(S2 \cdot S1) \cdot 2}$$

if textbook style is selected. Since we know that the function SQ( ) stands for $x^2$, we interpret the latter result as

$$\frac{S4^2}{2\cdot S3\cdot (S2\cdot S1)^2},$$

which indicates the position of the different stack input levels in the formula. By comparing this result with the original formula that we programmed, i.e.,

$$h_v = \frac{Q^2}{2g(by)^2},$$

we find that we must enter y in stack level 1 (S1), b in stack level 2 (S2), g in stack level 3 (S3), and Q in stack level 4 (S4).

## Prompt with an input string

These two approaches for identifying the order of the input data are not very efficient. You can, however, help the user identify the variables to be used by prompting him or her with the name of the variables. From the various methods provided by the User RPL language, the simplest is to use an input string and the function INPUT (↰ PRG NXT IN INPUT) to load your input data.

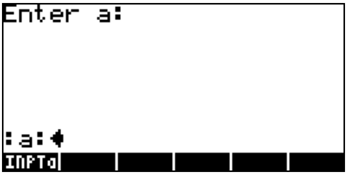
The following program prompts the user for the value of a variable a and places the input in stack level 1:

```
« “Enter a:“ {“↵:a:“ {2 0} V } INPUT OBJ→ »
```

This program includes the symbol :: (tag) and ↵(return), available through the keystroke combinations ↰ :: and ↱ ↵, both associated with the . key. The tag symbol (::) is used to label strings for input and output. The return symbol (↵) is similar to a carriage return in a computer. The strings between quotes (“ ”) are typed directly from the alphanumeric keyboard.

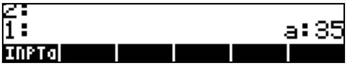
Save the program in a variable called INPTa (for INPuT a).

Try running the program by pressing the soft menu key labeled INPTa.



```
Enter a:



:a:◆
```

The result is a stack prompting the user for the value of a and placing the cursor right in front of the prompt :a: Enter a value for a, say 35, then press ENTER. The result is the input string `:a:35` in stack level 1.



```
2:
1:                    a:35
```

## A function with an input string

If you were to use this piece of code to calculate the function, f(a) = 2*a^2+3, you could modify the program to read as follows:

```
« “Enter a: “ {“↵:a: “ {2 0} V }
INPUT OBJ→ → a « ‘2*a^2+3‘ » »
```

Save this new program under the name ‘FUNCa’ (FUNCtion of a):

Run the program by pressing FUNCa. When prompted to enter the value of a enter, for example, 2, and press ENTER. The result is simply the algebraic $2a^2+3$, which is an incorrect result. The calculator provides functions for debugging programs to identify logical errors during program execution as shown below.

### Debugging the program

To figure out why it did not work we use the DBUG function in the calculator as follows:

| | |
|---|---|
| ' FUNCa ENTER | Copies program name to stack level 1 |
| ← PRG NXT NXT RUN DBG | Starts debugger |
| SST↓ | Step-by-step debugging, result: “Enter a:” |
| SST↓ | Result: {“ ↵ a:” {2 0} V} |
| SST↓ | Result: user is prompted to enter value of a |
| 2 ENTER | Enter a value of 2 for a. Result: “↵:a:2” |

| | |
|---|---|
| SST↓ | Result: a:2 |
| SST↓ | Result: empty stack, executing →a |
| SST↓ | Result: empty stack, entering subprogram « |
| SST↓ | Result: ‘2*a^2+3’ |
| SST↓ | Result: ‘2*a^2+3’, leaving subprogram » |
| SST↓ | Result: ‘2*a^2+3’, leaving main program» |

Further pressing the SST↓ soft menu key produces no more output since we have gone through the entire program, step by step. This run through the debugger did not provide any information on why the program is not calculating the value of $2a^2+3$ for a = 2. To see what is the value of a in the sub-program, we need to run the debugger again and evaluate a within the sub-program. Try the following:

| | |
|---|---|
| VAR | Recovers variables menu |
| ' FUNCa ENTER | Copies program name to stack level 1 |
| ← PRG NXT NXT RUN DBG | Starts debugger |
| SST↓ | Step-by-step debugging, result: “Enter a:” |
| SST↓ | Result: {“ ↵a:” {2 0} V} |
| SST↓ | Result: user is prompted to enter value of a |
| 2 ENTER | Enter a value of 2 for a. Result: “↵:a:2” |
| SST↓ | Result: a:2 |
| SST↓ | Result: empty stack, executing →a |
| SST↓ | Result: empty stack, entering subprogram « |

At this point we are within the subprogram « `‘2*a^2+3’` » which uses the local variable a. To see the value of a use:

| | |
|---|---|
| ALPHA ← A EVAL | This indeed shows that the local variable a = 2 |

Let’s kill the debugger at this point since we already know the result we will get. To kill the debugger press KILL. You receive an `<!> Interrupted` message acknowledging killing the debugger. Press ON to recover normal calculator display.

**Note**: In debugging mode, every time we press SST↓ the top left corner of the display shows the program step being executed. A soft key function called SST is also available under the RUN sub-menu within the PRG menu.

This can be used to execute at once any sub-program called from within a main program. Examples of the application of SST will be shown later.

### Fixing the program

The only possible explanation for the failure of the program to produce a numerical result seems to be the lack of the command →NUM after the algebraic expression '2*a^2+3'. Let's edit the program by adding the missing EVAL function. The program, after editing, should read as follows:

```
« "Enter a: " {"↵:a: " {2 0} V } INPUT
    OBJ→ → a « '2*a^2+3' →NUM » »
```

Store it again in variable FUNCa, and run the program again with a = 2. This time, the result is 11, i.e., $2*2^2+3 = 11$.

## Input string for two or three input values

In this section we will create a sub-directory, within the directory HOME, to hold examples of input strings for one, two, and three input data values. These will be generic input strings that can be incorporated in any future program, taking care of changing the variable names according to the needs of each program.

Let's get started by creating a sub-directory called PTRICKS (Programming TRICKS) to hold programming tidbits that we can later borrow from to use in more complex programming exercises. To create the sub-directory, first make sure that you move to the HOME directory. Within the HOME directory, use the following keystrokes to create the sub-directory PTRICKS:

| | |
|---|---|
| ' ALPHA ALPHA P T R I C K S ENTER | Enter directory name 'PTRICKS' |
| ← PRG MEM DIR CRDIR | Create directory |
| VAR | Recover variable listing |

A program may have more than 3 input data values. When using input strings we want to limit the number of input data values to 5 at a time for the simple reason that, in general, we have visible only 7 stack levels. If we use stack level 7 to give a title to the input string, and leave stack level 6 empty to facilitate reading the display, we have only stack levels 1 through 5 to define input variables.

**Input string program for two input values**

The input string program for two input values, say a and b, looks as follows:

```
« "Enter a and b: " {"↵:a:↵:b: " {2 0} V } INPUT OBJ→ »
```

This program can be easily created by modifying the contents of INPTa. Store this program into variable INPT2.

*Application*: evaluating a function of two variables

Consider the ideal gas law, pV = nRT, where p = gas pressure (Pa), V = gas volume(m$^3$), n = number of moles (gmol), R = universal gas constant = 8.31451_J/(gmol*K), and T = absolute temperature (K).

We can define the pressure p as a function of two variables, V and T, as p(V,T) = nRT/V for a given mass of gas since n will also remain constant. Assume that n = 0.2 gmol, then the function to program is

$$p(V,T) = 8.31451 \cdot 0.2 \cdot \frac{T}{V} = (1.662902\_\frac{J}{K}) \cdot \frac{T}{V}$$

We can define the function by typing the following program

```
« →V T '(1.662902_J/K)*(T/V)' »
```

and storing it into variable p.

The next step is to add the input string that will prompt the user for the values of V and T. To create this input stream, modify the program in p to read:

```
« "Enter V and T: " {"↵ :V:↵ :T: " {2 0} V }
INPUT OBJ→ → V T '(1.662902_J/K)*(T/V)' »
```

Store the new program back into variable p. Press p to run the program. Enter values of V = 0.01_m^3 and T = 300_K in the input string, then press ENTER. The result is 49887.06_J/m^3. The units of J/m^3 are equivalent to Pascals (Pa), the preferred pressure unit in the S.I. system.

> **Note**: because we deliberately included units in the function definition, the input values must have units attach to them in input to produce the proper result.

**Input string program for three input values**

The input string program for three input values, say a ,b, and c, looks as follows:

```
   « "Enter a, b and c: " {"↵ :a:↵ :b:↵ :c: " {2 0} V }
                         INPUT OBJ→ »
```

This program can be easily created by modifying the contents of INPT2 to make it look like shown immediately above. The resulting program can then be stored in a variable called INPT3. With this program we complete the collection of input string programs that will allow us to enter one, two, or three data values. Keep these programs as a reference and copy and modify them to fulfill the requirements of new programs you write.

***Application***: evaluating a function of three variables

Suppose that we want to program the ideal gas law including the number of moles, n, as an additional variable, i.e., we want to define the function

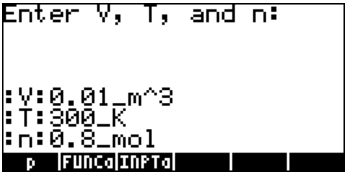
$$p(V,T,n) = (8.31451\_\frac{J}{K})\frac{n\cdot T}{V},$$

and modify it to include the three-variable input string. The procedure to put together this function is very similar to that used earlier in defining the function p(V,T). The resulting program will look like this:

```
« "Enter V, T, and n:" {" ↵ :V:↵ :T:↵ :n:" {2 0} V }
INPUT OBJ→ →V T n '(8.31451_J/(K*mol))*(n*T/V)'»
```

Store this result back into the variable p.To run the program, press p.

Enter values of V = 0.01_m^3, T = 300_K, and n = 0.8_mol. Before pressing ENTER, the stack will look like this:



```
Enter V, T, and n:


:V:0.01_m^3
:T:300_K
:n:0.8_mol
```

Press ENTER to get the result 199548.24_J/m^3, or 199548.24_Pa = 199.55 kPa.

## Input through input forms

Function INFORM (← PRG NXT IN INFOR.) can be used to create detailed input forms for a program. Function INFORM requires five arguments, in this order:

1. A title: a character string describing the input form
2. Field definitions: a list with one or more field definitions {$s_1$ $s_2$ … $s_n$}, where each field definition, $s_i$, can have one of two formats:
    a. A simple field label: a character string
    b. A list of specifications of the form {"label" "helpInfo" $type_0$ $type_1$ … $type_n$}. The "label" is a field label. The "helpInfo" is a character string describing the field label in detail, and the type specifications is a list of types of variables allowed for the field (see Chapter 24 for object types).
3. Field format information: a single number *col* or a list {*col tabs*}. In this specification, *col* is the number of columns in the input box, and *tabs* (optional) specifies the number of tab stops between the labels and the fields in the form. The list could be an empty list. Default values are *col* = 1 and *tabs* = 3.
4. List of reset values: a list containing the values to reset the different fields if the option RESET is selected while using the input form.
5. List of initial values: a list containing the initial values of the fields.

The lists in items 4 and 5 can be empty lists. Also, if no value is to be selected for these options you can use the NOVAL command (← PRG NXT IN NOVAL).

After function INFORM is activated you will get as a result either a zero, in case the CANCEL option is entered, or a list with the values entered in the fields in the order specified and the number 1, i.e., in the RPN stack:

```
2:    {v1 v2 … vn}
1:               1
```

Thus, if the value in stack level 1 is zero, no input was performed, while it this value is 1, the input values are available in stack level 2.

Example 1 - As an example, consider the following program, INFP1 (INput Form Program 1) to calculate the discharge Q in an open channel through Chezy's formula: $Q = C\cdot(R\cdot S)^{1/2}$, where C is the Chezy coefficient, a function of the channel surface's roughness (typical values 80-150), R is the hydraulic radius of the channel (a length), and S is the channel bed's slope (a dimensionless numbers, typically 0.01 to 0.000001). The following program defines an input form through function INFORM:

```
« " CHEZY'S EQN" { { "C:" "Chezy's coefficient" 0} { "R:"
"Hydraulic radius" 0 } { "S:" "Channel bed slope" 0} } { }
{ 120 1 .0001} { 110 1.5 .00001 }  INFORM »
```

In the program we can identify the 5 components of the input as follows:

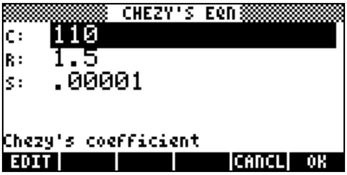
1. Title: `" CHEZY'S EQN"`
2. Field definitions: there are three of them, with labels "C:", "R:", "S:", info strings "Chezy coefficient", "Hydraulic radius", "Channel bed slope", and accepting only data type 0 (real numbers) for all of the three fields:

   ```
   { { "C:" "Chezy's coefficient" 0} { "R:" "Hydraulic
   radius" 0 } { "S:" "Channel bed slope" 0} }
   ```

3. Field format information: `{ }` (an empty list, thus, default values used)

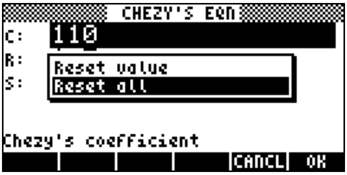
4. List of reset values: `{ 120 1 .0001}`
5. List of initial values: `{ 110 1.5 .00001}`

Save the program into variable INFP1. Press INFP1 to run the program. The input form, with initial values loaded, is as follows:

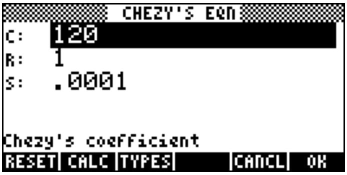


```
C:  110
R:  1.5
S:  .00001

Chezy's coefficient
```

To see the effect of resetting these values use NXT RESET (select *Reset all* to reset field values):

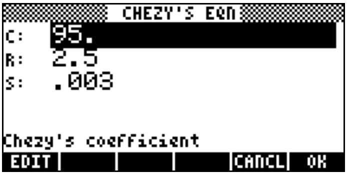


```
C:  110
R:
S:

Chezy's coefficient
```



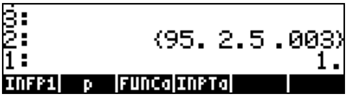
```
C:  120
R:  1
S:  .0001

Chezy's coefficient
```

Now, enter different values for the three fields, say, C = 95, R = 2.5, and S = 0.003, pressing OK after entering each of these new values. After these substitutions the input form will look like this:



```
C:  95.
R:  2.5
S:  .003

Chezy's coefficient
```

Now, to enter these values into the program press OK once more. This activates the function INFORM producing the following results in the stack:



```
3:
2:          {95. 2.5 .003}
1:                      1.
```

Thus, we demonstrated the use of function INFORM. To see how to use these input values in a calculation modify the program as follows:

```
« “ CHEZY’S EQN” { { “C:” “Chezy’s coefficient” 0} { “R:”
“Hydraulic radius” 0 } { “S:” “Channel bed slope” 0} } { }
{ 120 1 .0001} { 110 1.5 .00001 }  INFORM IF THEN OBJ→ DROP →
C R S ‘C*(R*S)’ →NUM “Q” →TAG ELSE “Operation cancelled”
MSGBOX END »
```

The program steps shown above after the INFORM command include a decision branching using the IF-THEN-ELSE-END construct (described in detail elsewhere in this Chapter). The program control can be sent to one of two possibilities depending on the value in stack level 1. If this value is 1 the control is passed to the commands:

```
OBJ→ DROP → C R S ‘C*√(R*S)’ →NUM “Q” →TAG
```

These commands will calculate the value of Q and put a tag (or label) to it. On the other hand, if the value in stack level 1 is 0 (which happens when a CANCL is entered while using the input box) , the program control is passed to the commands:

```
“Operation cancelled” MSGBOX
```

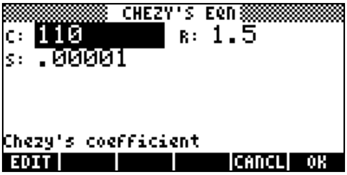
These commands will produce a message box indicating that the operation was cancelled.

> **Note:** Function MSGBOX belongs to the collection of output functions under the PRG/OUT sub-menu. Commands IF, THEN, ELSE, END are available under the PRG/BRCH/IF sub-menu. Functions OBJ→, →TAG are available under the PRG/TYPE sub-menu. Function DROP is available under the PRG/STACK menu. Functions → and →NUM are available in the keyboard.

Example 2 – To illustrate the use of item 3 (Field format information) in the arguments of function INFORM, change the empty list used in program INFP1 to { 2 1 }, meaning 2, rather than the default 3, columns, and only one tab stop between labels and values. Store this new program in variable INFP2:

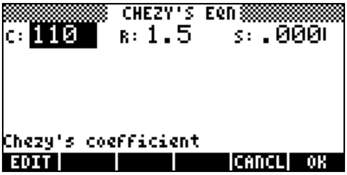
```
« " CHEZY'S EQN" { { "C:" "Chezy's coefficient" 0}
{ "R:" "Hydraulic radius" 0 } { "S:" "Channel bed slope"
0} } { 2 1 } { 120 1 .0001} { 110 1.5 .00001 }  INFORM
IF THEN OBJ→ DROP → C R S 'C*(R*S)' →NUM "Q" →TAG ELSE
"Operation cancelled" MSGBOX END »
```

Running program INFP2 produces the following input form:



```
CHEZY'S EQN
C: 110        R: 1.5
S: .00001

Chezy's coefficient
EDIT |  |  |  | CANCL | OK
```

Example 3 – Change the field format information list to { 3 0 }  and save the modified program into variable INFP3.  Run this program to see the new input form:



```
CHEZY'S EQN
C: 110   R: 1.5   S: .0001

Chezy's coefficient
EDIT |  |  |  | CANCL | OK
```

## Creating a choose box

Function CHOOSE (← PRG NXT IN CHOOS) allows the user to create a choose box in a program.  This function requires three arguments:

1. A prompt (a character string describing the choose box)
2. A list of choice definitions {$c_1$ $c_2$ … $c_n$}. A choice definition $c_i$ can have any of two formats:
   a. An object, e.g., a number, algebraic, etc., that will be displayed in the choose box and will also be the result of the choice.
   b. A list {object_displayed object_result} so that object_displayed is listed in the choose box, and object_result is selected as the result if this choice is selected.
3. A number indicating the position in the list of choice definitions of the default choice.  If this number is 0, no default choice is highlighted.

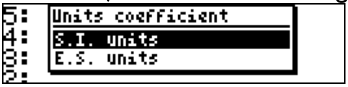
Activation of the CHOOSE function will return either a zero, if a CANCL action is used, or, if a choice is made, the choice selected (e.g., v) and the number 1, i.e., in the RPN stack:

```
2:            v
1:            1
```

Example 1 – Manning's equation for calculating the velocity in an open channel flow includes a coefficient, $C_u$, which depends on the system of units used. If using the S.I. (Systeme International), $C_u$ = 1.0, while if using the E.S. (English System), $C_u$ = 1.486. The following program uses a choose box to let the user select the value of $C_u$ by selecting the system of units. Save it into variable CHP1 (CHoose Program 1):

```
        « "Units coefficient" { { "S.I. units" 1}
            { "E.S. units" 1.486}  } 1 CHOOSE »
```

Running this program (press CHP1) shows the following choose box:



```
5: Units coefficient
4: S.I. units
3: E.S. units
2:
```

Depending on whether you select `S.I. units` or `E.S. units`, function CHOOSE placese either a value of 1 or a value of 1.486 in the stack. If you cancel the choose box, CHOICE returns a zero (0).

The values returned by function CHOOSE can be operated upon by other program commands as shown in the modified program CHP2:

```
« "Units coefficient" { { "S.I. units" 1} { "E.S. units"
1.486}  } 1 CHOOSE IF THEN "Cu" →TAG ELSE "Operation
cancelled" MSGBOX END »
```

The commands after the CHOOSE function in this new program indicate a decision based on the value of stack level 1 through the IF-THEN-ELSE-END construct. If the value in stack level 1 is 1, the commands `"Cu" →TAG` will produced a tagged result in the screen. If the value in stack level 1 is zero, the commands "Operation cancelled" MSGBOX will show a message box indicating that the operation was cancelled.

## Identifying output in programs

The simplest way to identify numerical program output is to "tag" the program results. A tag is simply a string attached to a number, or to any object. The string will be the name associated with the object. For example, earlier on, when debugging programs INTPa (or INPT1) and INPT2, we obtained as results tagged numerical output such as :a:35.

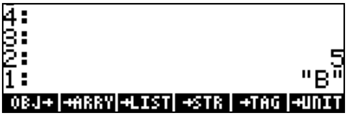
### Tagging a numerical result

To tag a numerical result you need to place the number in stack level 2 and the tagging string in stack level 2, then use the →TAG function (↰ PRG TYPE →TAG) For example, to produce the tagged result B:5., use:

5 ENTER ↱ "" ALPHA B ↰ PRG TYPE →TAG

### Decomposing a tagged numerical result into a number and a tag

To decompose a tagged result into its numerical value and its tag, simply use function OBJ→ (↰ PRG TYPE OBJ→). The result of decomposing a tagged number with →OBJ is to place the numerical value in stack level 2 and the tag in stack level 1. If you are interested in using the numerical value only, then you will drop the tag by using the backspace key ⬅. For example, decomposing the tagged quantity B:5 (see above), will produce:

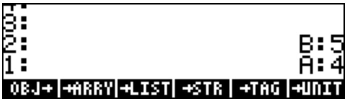


```
4:
3:
2:                5
1:              "B"
```

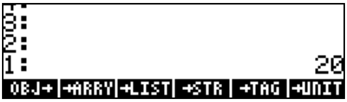
### "De-tagging" a tagged quantity

"De-tagging" means to extract the object out of a tagged quantity. This function is accessed through the keystroke combination: ↰ PRG TYPE NXT DTAG. For example, given the tagged quantity a:2, DTAG returns the numerical value 2.

**Note**: For mathematical operations with tagged quantities, the calculator will "detag" the quantity automatically before the operation. For example, the left-hand side figure below shows two tagged quantities before and after pressing the × key in RPN mode:



```
3:
2:                      B:5
1:                      A:4
```



```
3:
2:
1:                       20
```

## Examples of tagged output

Example 1 – tagging output from function FUNCa

Let's modify the function FUNCa, defined earlier, to produce a tagged output. Use ↱ FUNCa to recall the contents of FUNCa to the stack. The original function program reads

```
    « "Enter a: " {"↵:a: " {2 0} V } INPUT OBJ→ → a «
                    '2*a^2+3' →NUM » »
```

Modify it to read:

```
    « "Enter a: " {"↵:a: " {2 0} V } INPUT OBJ→ → a «
                '2*a^2+3' →NUM "F" →TAG » »
```

Store the program back into FUNCa by using ↰ FUNCa. Next, run the program by pressing FUNCa. Enter a value of 2 when prompted, and press ENTER. The result is now the tagged result `F:11`.

Example 2 – tagging input and output from function FUNCa

In this example we modify the program FUNCa so that the output includes not only the evaluated function, but also a copy of the input with a tag.

Use ↱ FUNCa to recall the contents of FUNCa to the stack:

```
    « "Enter a: " {"↵:a: " {2 0} V } INPUT OBJ→ → a «
                '2*a^2+3' →NUM "F" →TAG » »
```

Modify it to read:

```
    « "Enter a: " {"↵:a: " {2 0} V } INPUT OBJ→ → a «
             '2*a^2+3' EVAL "F" →TAG a SWAP» »
```

(Recall that the function SWAP is available by using ← PRG STACK SWAP). Store the program back into FUNCa by using ← FUNCa. Next, run the program by pressing FUNCa . Enter a value of 2 when prompted, and press ENTER. The result is now two tagged numbers `a:2.` in stack level 2, and `F:11.` in stack level 1.

**Note**: Because we use an input string to get the input data value, the local variable `a` actually stores a tagged value ( `:a:2`, in the example above). Therefore, we do not need to tag it in the output. All what we need to do is place an `a` before the SWAP function in the subprogram above, and the tagged input is placed in the stack. It should be pointed out that, in performing the calculation of the function, the tag of the tagged input a is dropped automatically, and only its numerical value is used in the calculation.

To see the operation of the function FUNCa, step by step, you could use the DBUG function as follows:

| | |
|---|---|
| ' FUNCa ENTER | Copies program name to stack level 1 |
| ← PRG NXT NXT RUN DBG | Starts debugger |
| SST↓ | Step-by-step debugging, result: "Enter a:" |
| SST↓ | Result: {" ↵a:" {2 0} V} |
| SST↓ | Result: user is prompted to enter value of a |
| 2 ENTER | Enter a value of 2 for a. Result: "↵a:2" |
| SST↓ | Result: a:2 |
| SST↓ | Result: empty stack, executing →a |
| SST↓ | Result: empty stack, entering subprogram « |
| SST↓ | Result: '2*a^2+3' |
| SST↓ | Result: empty stack, calculating |
| SST↓ | Result: 11., |
| SST↓ | Result: "F" |
| SST↓ | Result: F: 11. |
| SST↓ | Result: a:2. |
| SST↓ | Result: swap levels 1 and 2 |
| SST↓ | leaving subprogram » |
| SST↓ | leaving main program » |

Example 3 – tagging input and output from function p(V,T)

In this example we modify the program p so that the output tagged input values and tagged result. Use ↱ p to recall the contents of the program to the stack:

```
« "Enter V, T, and n:" {"↵ :V:↵ :T:↵ :n:" {2 0} V }
INPUT OBJ→ →V T n '(8.31451_J/(K*mol))*(n*T/V)' »
```

Modify it to read:

```
« "Enter V, T and n: " {"↵ :V:↵ :T: " {2 0} V } INPUT OBJ→
→V T n « V T n'(8.31451_J/(K*mol))*(n*T/V)' EVAL "p" →TAG
» »
```

**Note**: Notice that we have placed the calculation and tagging of the function p(V,T,n), preceded by a recall of the input variables V T n, into a sub-program [the sequence of instructions contained within the inner set of program symbols « » ]. This is necessary because without the program symbol separating the two listings of input variables (`V T N « V T n`), the program will assume that the input command

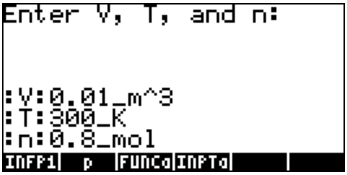
```
→V T N V T n
```

requires six input values, while only three are available. The result would have been the generation of an error message and the interruption of the program execution.

To include the subprogram mentioned above in the modified definition of program p, will require you to use ↱ _«» at the beginning and end of the sub-program. Because the program symbols occur in pairs, whenever ↱ _«» is invoked, you will need to erase the closing program symbol (») at the beginning, and the opening program symbol («) at the end, of the sub-program.

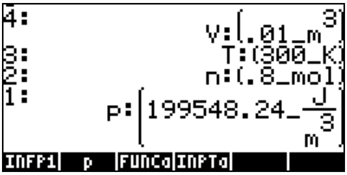
To erase any character while editing the program, place the cursor to the right of the character to be erased and use the backspace key ⬅.

Store the program back into variable p by using ← p. Next, run the program by pressing p. Enter values of V = 0.01_m^3, T = 300_K, and n = 0.8_mol, when prompted. Before pressing ENTER for input, the stack will look like this:



```
Enter V, T, and n:


:V:0.01_m^3
:T:300_K
:n:0.8_mol
```

After execution of the program, the stack will look like this:



```
4:          V:(.01_m^3)
3:             T:(300_K)
2:           n:(.8_mol)
1:  p:(199548.24_J/m^3)
```

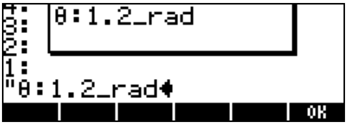
**In summary**: The common thread in the three examples shown here is the use of tags to identify input and output variables. If we use an input string to get our input values, those values are already pre-tagged and can be easily recall into the stack for output. Use of the →TAG command allows us to identify the output from a program.

## Using a message box

A message box is a fancier way to present output from a program. The message box command in the calculator is obtained by using ← PRG NXT OUT MSGBO. The message box command requires that the output string to be placed in the box be available in stack level 1. To see the operation of the MSGBOX command try the following exercise:

→ "" ALPHA → T ALPHA ← : : / • 2
→ _ ALPHA ← R ALPHA ← A ALPHA ← D
← PRG NXT OUT MSGBO

The result is the following message box:



```
4:  θ:1.2_rad
3:
2:
1:
"θ:1.2_rad◆
                          OK
```

Press OK to cancel the message box.

You could use a message box for output from a program by using a tagged output, converted to a string, as the output string for MSGBOX. To convert any tagged result, or any algebraic or non-tagged value, to a string, use the function →STR available at ← PRG TYPE →STR.

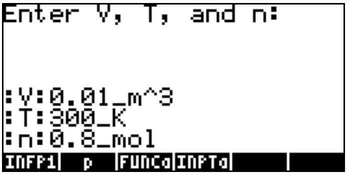
**Using a message box for program output**

The function p, from the last example, can be modified to read:

```
« "Enter V, T and n: " {"↵ :V:↵ :T:↵ :n: " {2 0} V }
INPUT OBJ→ → V T n « V T n
'(8.31451_J/(K*mol))*(n*T/V)' EVAL "p" →TAG →STR MSGBOX »
»
```

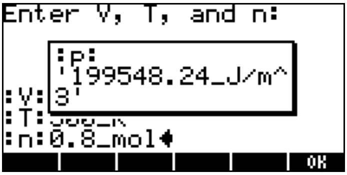
Store the program back into variable p by using ← p. Run the program by pressing p. Enter values of V = 0.01_m^3, T = 300_K, and n = 0.8_mol, when prompted.

As in the earlier version of p, before pressing ENTER for input, the stack will look like this:



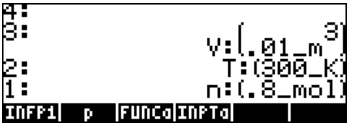
```
Enter V, T, and n:


:V:0.01_m^3
:T:300_K
:n:0.8_mol
INPT1   p   FUNCa INPTa
```

The first program output is a message box containing the string:



```
Enter V, T, and n:
:p:
'199548.24_J/m^
3'
:V:
:T:
:n:0.8_mol
```

Press OK to cancel message box output.  The stack will now look like this:



```
4:
3:
              V:(.01_m^3)
2:             T:(300_K)
1:            n:(.8_mol)
```

### Including input and output in a message box

We could modify the program so that not only the output, but also the input, is included in a message box.  For the case of program p, the modified program will look like:

```
« "Enter V, T and n: " {"↵ :V:↵ :T:↵ :n: " {2 0} V }
INPUT OBJ→ → V T n « V →STR "↵ " + T →STR "↵ " + n →STR
"↵ " + '(8.31451_J/(K*mol))*(n*T/V)' EVAL "p" →TAG →STR +
+ + MSGBOX » »
```

Notice that you need to add the following piece of code after each of the variable names V, T, and n, within the sub-program:

```
→STR "↵ " +
```

To get this piece of code typed in the first time use:

⬅ PRG TYPE →STR ➡ "" ➡ ↵ ▶ +

Because the functions for the TYPE menu remain available in the soft menu keys, for the second and third occurrences of the piece of code (→STR "↵ " + ) within the sub-program (i.e., after variables T and n, respectively), all you need to use is:

→STR ➡ "" ➡ ↵ ▶ +

You will notice that after typing the keystroke sequence ⮣ ⟵ a new line is generated in the stack.

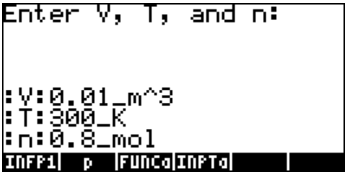
The last modification that needs to be included is to type in the plus sign three times after the call to the function at the very end of the sub-program.

**Note**: The plus sign (+) in this program is used to *concatenate* strings. *Concatenation* is simply the operation of joining individual character strings.

To see the program operating:

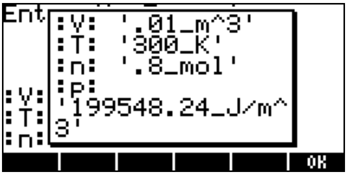
- Store the program back into variable p by using ⮢ ▦.
- Run the program by pressing ▦.
- Enter values of V = 0.01_m^3, T = 300_K, and n = 0.8_mol, when prompted.

As in the earlier version of [ p ], before pressing [ENTER] for input, the stack will look like this:



```
Enter V, T, and n:

:V:0.01_m^3
:T:300_K
:n:0.8_mol
```

The first program output is a message box containing the string:



```
:V: '.01_m^3'
:T: '300_K'
:n: '.8_mol'
:p:
'199548.24_J/m^
3'
```

Press ▦ to cancel message box output.

**Incorporating units within a program**

As you have been able to observe from all the examples for the different versions of program ▦ presented in this chapter, attaching units to input values may be a tedious process. You could have the program itself attach those units to the input and output values. We will illustrate these options by modifying yet once more the program ▦, as follows.

Recall the contents of program ▦ to the stack by using ↱▦, and modify them to look like this:

**Note**: I've separated the program arbitrarily into several lines for easy reading. This is not necessarily the way that the program shows up in the calculator's stack. The sequence of commands is correct, however. Also, recall that the character ↵ does not show in the stack, instead it produces a new line.

```
« "Enter V,T,n [S.I.]: " {"↵ :V:↵ :T:↵ :n: " {2 0} V }
INPUT OBJ→ →V T n
« V '1_m^3'* T '1_K' * n '1_mol' * →V T n
« V "V" →TAG →STR "↵ "+ T "T" →TAG →STR "↵ "+ n
"n" →TAG →STR "↵ "+
'(8.31451_J/(K*mol))*(n*T/V)' EVAL "p" →TAG →STR + + +
MSGBOX » » »
```

This new version of the program includes an additional level of sub-programming (i.e., a third level of program symbols « », and some steps using lists, i.e.,

```
    V '1_m^3'* { }+ T '1_K' * + n '1_mol' *+ EVAL →V T n
```

The *interpretation* of this piece *of code* is as follows. (We use input string values of `:V:0.01`, `:T:300`, and `:n:0.8`):

1. `V` : The value of V, as a tagged input (e.g., V:0.01) is placed in the stack.

2. `'1_m^3'` : The S.I. units corresponding to V are then placed in stack level 1, the tagged input for V is moved to stack level 2.

3. `*` : By multiplying the contents of stack levels 1 and 2,

we generate a number with units (e.g., 0.01_m^3), but the tag is lost.

4. `T '1_K' *` :Calculating value of T including S.I. units

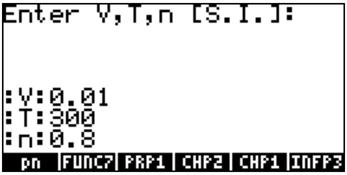
5. `n '1_mol' *` : Calculating value of n including units

6. `→V T n` : The values of V, T, and n, located respectively in stack levels 3, 2, and 1, are passed on to the next level of sub-programming.

To see this version of the program in action do the following:

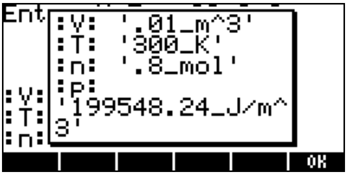
- Store the program back into variable p by using [←][ p ].
- Run the program by pressing [ p ].
- Enter values of V = 0.01, T = 300, and n = 0.8, when prompted (no units required now).

Before pressing ENTER for input, the stack will look like this:



```
Enter V,T,n [S.I.]:


:V:0.01
:T:300
:n:0.8
```

Press ENTER to run the program. The output is a message box containing the string:



```
:V: '.01_m^3'
:T: '300_K'
:n: '.8_mol'
:p:
'199548.24_J/m^
3'
```

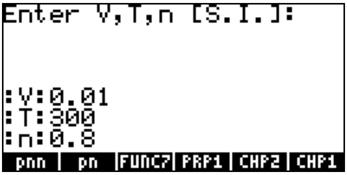
Press OK to cancel message box output.

### Message box output without units

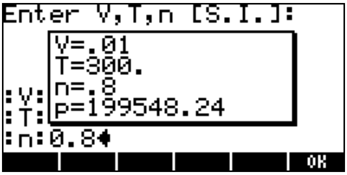
Let's modify the program p once more to eliminate the use of units throughout it. The unit-less program will look like this:

```
« "Enter V,T,n [S.I.]: " {"↵ :V:↵ :T:↵ :n: " {2 0} V }
INPUT OBJ→ →V T n
« V DTAG  T DTAG  n DTAG →  V T n
« "V="  V →STR + "↵ "+  "T="  T  →STR + "↵ "+ "n="   n →STR +
"↵ "+
'8.31451*n*T/V' EVAL →STR "p=" SWAP + + + + MSGBOX » » »
```

And when run with the input data V = 0.01, T = 300, and n = 0.8, produces the message box output:


Enter V,T,n [S.I.]:

:V:0.01
:T:300
:n:0.8


Enter V,T,n [S.I.]:
V=.01
T=300.
n=.8
p=199548.24
:V:
:T:
:n:0.8

Press OK to cancel the message box output.

## Relational and logical operators

So far we have worked mainly with sequential programs. The User RPL language provides statements that allow branching and looping of the program flow. Many of these make decisions based on whether a logical statement is true or not. In this section we present some of the elements used to construct such logical statements, namely, relational and logical operators.

### Relational operators

Relational operators are those operators used to compare the relative position of two objects. For example, dealing with real numbers only, relational operators are used to make a statement regarding the relative position of two or more real numbers. Depending on the actual numbers used, such a statement can be true (represented by the numerical value of 1. in the calculator), or false (represented by the numerical value of 0. in the calculator).

The relational operators available for programming the calculator are:

| Operator | Meaning | Example |
|---|---|---|
| == | "is equal to" | 'x==2' |
| ≠ | "is not equal to" | '3 ≠ 2' |
| < | "is less than" | 'm<n' |
| > | "is greater than" | '10>a' |
| ≥ | "is greater than or equal to" | 'p ≥ q' |
| ≤ | "is less than or equal to" | '7≤12' |

All of the operators, except == (which can be created by typing ↱ = ↱ = ), are available in the keyboard. They are also available in ↰ PRG TEST.

Two numbers, variables, or algebraics connected by a relational operator form a logical expression that can take value of true (1.), false (0.), or could simply not be evaluated. To determine whether a logical statement is true or not, place the statement in stack level 1, and press EVAL (EVAL). Examples:

'2<10' EVAL, result: 1. (true)

'2>10' EVAL, result: 0. (false)

In the next example it is assumed that the variable m is not initialized (it has not been given a numerical value):

'2==m' EVAL, result: '2==m'

The fact that the result from evaluating the statement is the same original statement indicates that the statement cannot be evaluated uniquely.

## Logical operators

Logical operators are logical particles that are used to join or modify simple logical statements. The logical operators available in the calculator can be easily accessed through the keystroke sequence: ↰ PRG TEST NXT.

The available logical operators are: AND, OR, XOR (exclusive or), NOT, and SAME. The operators will produce results that are true or false, depending on the truth-value of the logical statements affected. The operator NOT (negation) applies to a single logical statements. All of the others apply to two logical statements.

Tabulating all possible combinations of one or two statements together with the resulting value of applying a certain logical operator produces what is called the truth table of the operator. The following are truth tables of each of the standard logical operators available in the calculator:

| p | NOT p |
|---|---|
| 1 | 0 |
| 0 | 1 |

| p | q | p AND q |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

| p | q | p OR q |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

| p | q | p XOR q |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

The calculator includes also the logical operator SAME. This is a non-standard logical operator used to determine if two objects are identical. If they are identical, a value of 1 (true) is returned, if not, a value of 0 (false) is returned. For example, the following exercise, in RPN mode, returns a value of 0:

‘SQ(2)’ ENTER 4 ENTER SAME

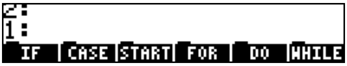
Please notice that the use of SAME implies a very strict interpretation of the word “identical.” For that reason, SQ(2) is not identical to 4, although they both evaluate, numerically, to 4.

## Program branching

Branching of a program flow implies that the program makes a decision among two or more possible flow paths. The User RPL language provides a number of commands that can be used for program branching. The menus containing these commands are accessed through the keystroke sequence:

← PRG BRCH

This menu shows sub-menus for the program constructs



The program constructs IF…THEN..ELSE…END, and CASE…THEN…END will be referred to as program branching constructs. The remaining constructs, namely, START, FOR, DO, and WHILE, are appropriate for controlling repetitive processing within a program and will be referred to as program loop constructs. The latter types of program constructs are presented in more detail in a later section.

### Branching with IF

In this section we presents examples using the constructs IF…THEN…END and IF…THEN…ELSE…END.

### The IF…THEN…END construct

The IF…THEN…END is the simplest of the IF program constructs. The general format of this construct is:

```
    IF logical_statement THEN program_statements END.
```

The operation of this construct is as follows:

1. Evaluate logical_statement.
2. If logical_statement is true, perform program _statements and continue program flow after the END statement.
3. If logical_statement is false, skip program_statements and continue program flow after the END statement.

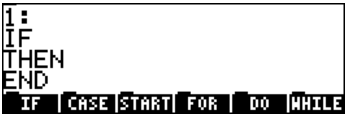
To type in the particles IF, THEN, ELSE, and END, use:

← PRG BRCH IF

The functions IF THEN ELSE END are available in that menu to be typed selectively by the user. Alternatively, to produce an IF…THEN…END construct directly on the stack, use:

← PRG BRCH ← IF

This will create the following input in the stack:



```
1:
IF
THEN
END
```

With the cursor ⬅ in front of the IF statement prompting the user for the logical statement that will activate the IF construct when the program is executed.
Example: Type in the following program:

```
« → x « IF 'x<3' THEN 'x^2' EVAL END "Done" MSGBOX » »
```

and save it under the name 'f1'. Press VAR and verify that variable f1 is indeed available in your variable menu. Verify the following results:

0 f1 Result: 0 1.2 f1 Result: 1.44
3.5 f1 Result: no action 10 f1 Result: no action

These results confirm the correct operation of the IF…THEN…END construct. The program, as written, calculates the function $f_1(x) = x^2$, if $x < 3$ (and not output otherwise).

**The IF…THEN…ELSE…END construct**

The IF…THEN…ELSE…END construct permits two alternative program flow paths based on the truth value of the logical_statement. The general format of this construct is:

```
IF logical_statement THEN program_statements_if_true
          ELSE program_statements_if_false END.
```

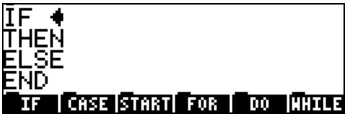
The operation of this construct is as follows:

1. Evaluate logical_statement.
2. If logical_statement is true, perform program statements_if_true and continue program flow after the END statement.
3. If logical_statement is false, perform program statements_if_false and continue program flow after the END statement.

To produce an IF…THEN…ELSE…END construct directly on the stack, use:

← PRG BRCH → IF

This will create the following input in the stack:



```
IF ◆
THEN
ELSE
END
```

Example: Type in the following program:

```
« → x « IF 'x<3' THEN 'x^2' ELSE '1-x' END EVAL "Done" MSGBOX
» »
```

and save it under the name 'f2'. Press VAR and verify that variable f2 is indeed available in your variable menu. Verify the following results:

| | |
|---|---|
| 0 f2 Result: 0 | 1.2 f2 Result: 1.44 |
| 3.5 f2 Result: -2.5 | 10 f2 Result: -9 |

These results confirm the correct operation of the IF…THEN…ELSE…END construct. The program, as written, calculates the function

$$f_2(x) = \begin{cases} x^2, if\ x < 3 \\ 1 - x, otherwise \end{cases}$$

**Note**: For this particular case, a valid alternative would have been to use an IFTE function of the form: 'f2(x) = IFTE(x<3,x^2,1-x)'

### Nested IF…THEN…ELSE…END constructs

In most computer programming languages where the IF…THEN…ELSE…END construct is available, the general format used for program presentation is the following:

```
IF logical_statement THEN
      program_statements_if_true
ELSE
      program_statements_if_false
END
```

In designing a calculator program that includes IF constructs, you could start by writing by hand the pseudo-code for the IF constructs as shown above. For example, for program f2, you could write

```
IF $x<3$ THEN
      $x^2$
ELSE
      $1-x$
END
```

While this simple construct works fine when your function has only two branches, you may need to nest IF…THEN…ELSE…END constructs to deal with function with three or more branches. For example, consider the function

$$f_3(x) = \begin{cases} x^2, \, if \;\; x < 3 \\ 1 - x, \;\; if \; 3 \le x < 5 \\ \sin(x), \, if \;\; 5 \le x < 3\pi \\ \exp(x), if \;\; 3\pi \le x < 15 \\ -2, \, elsewhere \end{cases}$$

Here is a possible way to evaluate this function using IF… THEN … ELSE … END constructs:

```
IF x<3 THEN
        x^2
ELSE
        IF x<5 THEN
                1-x
        ELSE
                IF x<3π         THEN
                        sin(x)
                ELSE
                        IF x<15 THEN
                                exp(x)
                        ELSE
                                -2
                        END
                END
        END
END
```

A complex IF construct like this is called a set of *nested* IF … THEN … ELSE … END constructs.

A possible way to evaluate f3(x), based on the nested IF construct shown above, is to write the program:

```
« → x « IF ‘x<3‘ THEN ‘x^2‘ ELSE IF ‘x<5‘ THEN ‘1-x‘ ELSE IF
‘x<3*π‘ THEN ‘SIN(x)‘ ELSE IF ‘x<15‘ THEN ‘EXP(x)‘ ELSE –2
END END END END EVAL » »
```

Store the program in variable f3 and try the following evaluations:

1.5 f3 *Result*: 2.25 (i.e., $x^2$)
2.5 f3 *Result*: 6.25 (i.e., $x^2$)
4.2 f3 *Result*: -3.2 (i.e., 1-x)
5.6 f3 *Result*: -0.631266... (i.e., sin(x), with x in radians)
12 f3 *Result*: 162754.791419 (i.e., exp(x))
23 f3 *Result*: -2. (i.e., -2)

## The CASE construct

The CASE construct can be used to code several possible program flux paths, as in the case of the nested IF constructs presented earlier. The general format of this construct is as follows:

```
CASE
Logical_statement1 THEN program_statements1 END
Logical_statement2 THEN program_statements2 END
.
.
.
Logical_statement THEN program_statements END
Default_program_statements (optional)
END
```

When evaluating this construct, the program tests each of the *logical_statements* until it finds one that is true. The program executes the corresponding *program_statements*, and passes program flow to the statement following the END statement.

The CASE, THEN, and END statements are available for selective typing by using ← PRG BRCH CASE.

If you are in the BRCH menu, i.e., (↰ PRG BRCH) you can use the following shortcuts to type in your CASE construct (The location of the cursor is indicated by the symbol ⬅ ):

- ↰ CASE: Starts the case construct providing the prompts: CASE ⬅ THEN END END

- ↱ CASE: Completes a CASE line by adding the particles THEN ⬅ END

Example – program $f_3(x)$ using the CASE statement
The function is defined by the following 5 expressions:

$$f_3(x) = \begin{cases} x^2, \mathit{if}\ x < 3 \\ 1 - x,\ \mathit{if}\ 3 \le x < 5 \\ \sin(x), \mathit{if}\ 5 \le x < 3\pi \\ \exp(x), \mathit{if}\ 3\pi \le x < 15 \\ -2, \mathit{elsewhere} \end{cases}$$

Using the CASE statement in User RPL language we can code this function as:

```
« → x « CASE 'x<3' THEN 'x^2' END 'x<5' THEN '1-x' END
'x<3*π' THEN 'SIN(x)' END 'x<15' THEN 'EXP(x)' END -2 END
EVAL » »
```

Store the program into a variable called f3c. Then, try the following exercises:

| | | |
|---|---|---|
| 1.5 | f3c | *Result*: 2.25 (i.e., $x^2$) |
| 2.5 | f3c | *Result*: 6.25 (i.e., $x^2$) |
| 4.2 | f3c | *Result*: -3.2 (i.e., 1-x) |
| 5.6 | f3c | *Result*: -0.631266… (i.e., sin(x), with x in radians) |
| 12 | f3c | *Result*: 162754.791419 (i.e., exp(x)) |
| 23 | f3c | *Result*: -2. (i.e., -2) |

As you can see, f3c produces exactly the same results as f3. The only difference in the programs is the branching constructs used. For the case of function $f_3(x)$, which requires five expressions for its definition, the CASE construct may be easier to code than a number of nested IF … THEN … ELSE … END constructs.

## Program loops

Program loops are constructs that permit the program the execution of a number of statements repeatedly. For example, suppose that you want to calculate the summation of the square of the integer numbers from 0 to n, i.e.,

$$S = \sum_{k=0}^{n} k^2$$

To calculate this summation all that you have to do is use the ↱ Σ key within the equation editor and load the limits and expression for the summation (examples of summations are presented in Chapters 2 and 13). However, in order to illustrate the use of programming loops, we will calculate this summation with our own User RPL codes. There are four different commands that can be used to code a program loop in User RPL, these are START, FOR, DO, and WHILE. The commands START and FOR use an index or counter to determine how many times the loop is executed. The commands DO and WHILE rely on a logical statement to decide when to terminate a loop execution. Operation of the loop commands is described in detail in the following sections.

### The START construct

The START construct uses two values of an index to execute a number of statements repeatedly. There are two versions of the START construct: START…NEXT and START … STEP. The START…NEXT version is used when the index increment is equal to 1, and the START…STEP version is used when the index increment is determined by the user.

Commands involved in the START construct are available through:

↰ PRG BRCH START START

Within the BRCH menu (↰ PRG BRCH) the following keystrokes are available to generate START constructs (the symbol indicates cursor position):

- ↰ START : Starts the START…NEXT construct: START ⬅ NEXT

- ↱ START : Starts the START…STEP construct: START ⬅ STEP

**The START…NEXT construct**

The general form of this statement is:

```
start_value end_value START program_statements NEXT
```

Because for this case the increment is 1, in order for the loop to end you should ensure that `start_value < end_value`. Otherwise you will produce what is called an <u>infinite (never-ending) loop.</u>

Example – calculating of the summation S defined above

The START…NEXT construct contains an index whose value is inaccessible to the user. Since for the calculation of the sum the index itself (k, in this case) is needed, we must create our own index, k, that we will increment within the loop each time the loop is executed. A possible implementation for the calculation of S is the program:

```
« 0. DUP →n S k « 0. n START k SQ S + 1. ‘k‘ STO+ ‘S‘ STO
NEXT S “S” →TAG » »
```

Type the program in, and save it in a variable called S1.

Here is a brief explanation of how the program works:

1. This program needs an integer number as input. Thus, before execution, that number (n) is in stack level 1. The program is then executed.

2. A zero is entered, moving n to stack level 2.
3. The command DUP, which can be typed in as ALPHA ALPHA D U P ALPHA, copies the contents of stack level 1, moves all the stack levels upwards, and places the copy just made in stack level 1. Thus, after DUP is executed, n is in stack level 3, and zeroes fill stack levels 1 and 2.
4. The piece of code `→ n S k` stores the values of n, 0, and 0, respectively into local variables n, S, k. We say that the variables n, S, and k have been <u>initialized </u>(S and k to zero, n to whatever value the user chooses).
5. The piece of code `0. n START` identifies a START loop whose index will take values of 0, 1, 2, …, n
6. The sum S is incremented by $k^2$ in the piece of code that reads: `k SQ S +`
7. The index k is incremented by 1 in the piece of code that reads: `1. k +`
8. At this point, the updated values of S and k are available in stack levels 2 and 1, respectively. The piece of code `'k' STO` stores the value from stack level 1 into local variable k. The updated value of S now occupies stack level 1.
9. The piece of code `'S' STO` stores the value from stack level 1 into local variable k. The stack is now empty.
10. The particle NEXT increases the index by one and sends the control to the beginning of the loop (step 6).
11. The loop is repeated until the loop index reaches the maximum value, n.
12. The last part of the program recalls the last value of S (the summation), tags it, and places it in stack level 1 to be viewed by the user as the program output.

To see the program in action, step by step, you can use the debugger as follows (use n = 2). Let SL1 mean stack level 1:

| | |
|---|---|
| VAR 2 ['] S1 ENTER | Place a 2 in level 2, and the program name, 'S1', in level 1 |
| ← PRG NXT NXT RUN DBG | Start the debugger. SL1 = 2. |
| SST↓ | SL1 = 0., SL2 = 2. |
| SST↓ | SL1 = 0., SL2 = 0., SL3 = 2. (DUP) |
| SST↓ | Empty stack (-> n S k) |
| SST↓ | Empty stack (« - start subprogram) |

| | |
|---|---|
| SST↓ | SL1 = 0., (start value of loop index) |
| SST↓ | SL1 = 2.(n), SL2 = 0. (end value of loop index) |
| SST↓ | Empty stack (START – beginning of loop) |

-- loop execution number 1 for k = 0

| | |
|---|---|
| SST↓ | SL1 = 0. (k) |
| SST↓ | SL1 = 0. (SQ(k) = $k^2$) |
| SST↓ | SL1 = 0.(S), SL2 = 0. ($k^2$) |
| SST↓ | SL1 = 0. (S + $k^2$) |
| SST↓ | SL1 = 1., SL2 = 0. (S + $k^2$) |
| SST↓ | SL1 = 0.(k), SL2 = 1., SL3 = 0. (S + $k^2$) |
| SST↓ | SL1 = 1.(k+1), SL2 = 0. (S + $k^2$) |
| SST↓ | SL1 = 'k', SL2 = 1., SL3 = 0. (S + $k^2$) |
| SST↓ | SL1 = 0. (S + $k^2$) [Stores value of SL2 = 1, into SL1 = 'k'] |
| SST↓ | SL1 = 'S', SL2 = 0. (S + $k^2$) |
| SST↓ | Empty stack [Stores value of SL2 = 0, into SL1 = 'S'] |
| SST↓ | Empty stack (NEXT – end of loop) |

-- loop execution number 2 for k = 1

| | |
|---|---|
| SST↓ | SL1 = 1. (k) |
| SST↓ | SL1 = 1. (SQ(k) = $k^2$) |
| SST↓ | SL1 = 0.(S), SL2 = 1. ($k^2$) |
| SST↓ | SL1 = 1. (S + $k^2$) |
| SST↓ | SL1 = 1., SL2 = 1. (S + $k^2$) |
| SST↓ | SL1 = 1.(k), SL2 = 1., SL3 = 1. (S + $k^2$) |
| SST↓ | SL1 = 2.(k+1), SL2 = 1. (S + $k^2$) |
| SST↓ | SL1 = 'k', SL2 = 2., SL3 = 1. (S + $k^2$) |
| SST↓ | SL1 = 1. (S + $k^2$) [Stores value of SL2 = 2, into SL1 = 'k'] |
| SST↓ | SL1 = 'S', SL2 = 1. (S + $k^2$) |
| SST↓ | Empty stack [Stores value of SL2 = 1, into SL1 = 'S'] |

SST↓ Empty stack (NEXT – end of loop)

-- loop execution number 3 for k = 2

SST↓ SL1 = 2. (k)
SST↓ SL1 = 4. (SQ(k) = $k^2$)
SST↓ SL1 = 1.(S), SL2 = 4. ($k^2$)
SST↓ SL1 = 5. (S + $k^2$)
SST↓ SL1 = 1., SL2 = 5. (S + $k^2$)
SST↓ SL1 = 2.(k), SL2 = 1., SL3 = 5. (S + $k^2$)
SST↓ SL1 = 3.(k+1), SL2 = 5. (S + $k^2$)
SST↓ SL1 = 'k', SL2 = 3., SL3 = 5. (S + $k^2$)
SST↓ SL1 = 5. (S + $k^2$) [Stores value of SL2 = 3, into SL1 = 'k']
SST↓ SL1 = 'S', SL2 = 5. (S + $k^2$)
SST↓ Empty stack [Stores value of SL2 = 0, into SL1 = 'S']
SST↓ Empty stack (NEXT – end of loop)

-- for n = 2, the loop index is exhausted and control is passed to the statement following NEXT

SST↓ SL1 = 5 (S is recalled to the stack)
SST↓ SL1 = "S", SL2 = 5 ("S" is placed in the stack)
SST↓ SL1 = S:5 (tagging output value)
SST↓ SL1 = S:5 (leaving sub-program »)
SST↓ SL1 = S:5 (leaving main program »)

The step-by-step listing is finished. The result of running program S1 with n = 2, is S:5.

Check also the following results: VAR

| | | | |
|---|---|---|---|
| 3 S1 | Result: `S:14` | 4 S1 | Result: `S:30` |
| 5 S1 | Result: `S:55` | 8 S1 | Result: `S:204` |
| 10 S1 | Result: `S:385` | 20 S1 | Result: `S:2870` |
| 30 S1 | Result: `S:9455` | 100 S1 | Result: `S:338350` |

### The START…STEP construct

The general form of this statement is:

```
start_value end_value START program_statements increment
NEXT
```

The start_value, end_value, and `increment` of the loop index can be positive or negative quantities. For `increment > 0`, execution occurs as long as the index is less than or equal to `end_value`. For `increment < 0`, execution occurs as long as the index is greater than or equal to `end_value`.

**Example** – generating a list of values

Suppose that you want to generate a list of values of x from x = 0.5 to x = 6.5 in increments of 0.5. You can write the following program:

```
« → xs xe dx « xs DUP xe START DUP dx + dx STEP DROP xe
xs – dx / ABS 1 + →LIST » »
```

and store it in variable GLIST.

In this program , xs = starting value of the loop, xe = ending value of the loop, dx = increment value for loop. The program places values of xs, xs+dx, xs+2·dx, xs+3·dx, … in the stack. Then, it calculates the number of elements generated using the piece of code: `xe xs – dx / ABS 1. +`

Finally, the program puts together a list with the elements placed in the stack.

- Check out that the program call 0.5 ENTER 2.5 ENTER 0.5 ENTER GLIST produces the list {0.5 1. 1.5 2. 2.5}.
- To see step-by-step operation use the program DBUG for a short list, for example:

| | |
|---|---|
| VAR 1 SPC 1.5 SPC 0.5 ENTER | Enter parameters 1 1.5 0.5 |
| [ ' ] GLIST ENTER | Enter the program name in level 1 |
| ← PRG NXT NXT RUN DBG | Start the debugger. |

Use SST↓ to step into the program and see the detailed operation of each command.

## The FOR construct

As in the case of the START command, the FOR command has two variations: the FOR…NEXT construct, for loop index increments of 1, and the FOR…STEP construct, for loop index increments selected by the user. Unlike the START command, however, the FOR command does require that we provide a name for the loop index (e.g., j, k, n). We need not to worry about incrementing the index ourselves, as done in the examples using START. The value corresponding to the index is available for calculations.

Commands involved in the FOR construct are available through:

↰ PRG BRCH FOR

Within the BRCH menu (↰ PRG BRCH) the following keystrokes are available to generate FOR constructs (the symbol ⬅ indicates cursor position):

- ↰ FOR: Starts the FOR…NEXT construct: FOR ⬅ NEXT
- ↱ FOR: Starts the FOR…STEP construct: FOR ⬅ STEP

### The FOR…NEXT construct

The general form of this statement is:

```
start_value end_value FOR loop_index program_statements
NEXT
```

To avoid an infinite loop, make sure that `start_value < end_value`.

**Example** – calculate the summation S using a FOR…NEXT construct
The following program calculates the summation

$$S = \sum_{k=0}^{n} k^2$$

Using a FOR…NEXT loop:

```
« 0 →n S « 0 n FOR k k SQ S + ‘S‘ STO NEXT S “S” →TAG » »
```

Store this program in a variable S2. Verify the following exercises: VAR

| | | | |
|---|---|---|---|
| 3 S2 | Result: `S:14` | 4 S2 | Result: `S:30` |
| 5 S2 | Result: `S:55` | 8 S2 | Result: `S:204` |
| 10 S2 | Result: `S:385` | 20 S2 | Result: `S:2870` |
| 30 S2 | Result: `S:9455` | 100 S2 | Result: `S:338350` |

You may have noticed that the program is much simpler than the one stored in S1. There is no need to initialize k, or to increment k within the program. The program itself takes care of producing such increments.

**The FOR…STEP construct**

The general form of this statement is:

```
start_value end_value FOR loop_index program_statements
increment STEP
```

The start_value, end_value, and `increment` of the loop index can be positive or negative quantities. For `increment > 0`, execution occurs as long as the index is less than or equal to `end_value`. For `increment < 0`, execution occurs as long as the index is greater than or equal to `end_value`. Program statements are executed at least once (e.g., `1 0 START 1 1 STEP` returns 1)

<u>Example</u> – generate a list of numbers using a FOR…STEP construct
Type in the program:

```
« → xs xe dx « xe xs – dx / ABS 1. + → n « xs xe FOR x
x dx STEP n →LIST » » »
```

and store it in variable GLIST.

- Check out that the program call 0.5 ENTER 2.5 ENTER 0.5 ENTER GLIS2 produces the list {0.5 1. 1.5 2. 2.5}.
- To see step-by-step operation use the program DBUG for a short list, for example:

| | |
|---|---|
| VAR 1 SPC 1.5 SPC 0.5 ENTER | Enter parameters 1 1.5 0.5 |
| [‘] GLIS2 ENTER | Enter the program name in level 1 |
| ← PRG NXT NXT RUN DBG | Start the debugger. |

Use SST↓ to step into the program and see the detailed operation of each command.

## The DO construct

The general structure of this command is:

```
DO program_statements UNTIL logical_statement END
```

The DO command starts an indefinite loop executing the program_statements until the logical_statement returns FALSE (0). The `logical_statement` must contain the value of an index whose value is changed in the `program_statements`.

Example 1 - This program produces a counter in the upper left corner of the screen that adds 1 in an indefinite loop until a keystroke (press any key) stops the counter: `« 0 DO DUP 1 DISP 1 + UNTIL KEY END DROP »`

Command KEY evaluates to TRUE when a keystroke occurs.

Example 2 – calculate the summation S using a DO…UNTIL…END construct
The following program calculates the summation

$$S = \sum_{k=0}^{n} k^2$$

Using a DO…UNTIL…END loop:

```
« 0. → n S « DO n SQ S + ‘S‘ STO n 1 – ‘n‘ STO UNTIL
‘n<0‘ END S “S” →TAG » »
```

Store this program in a variable SSQ3. Verify the following exercises: VAR

| | | | | |
|---|---|---|---|---|
| 3 SSQ3 | Result: `S:14` | | 4 SSQ3 | Result: `S:30` |
| 5 SSQ3 | Result: `S:55` | | 8 SSQ3 | Result: `S:204` |
| 10 SSQ3 | Result: `S:385` | | 20 SSQ3 | Result: `S:2870` |
| 30 SSQ3 | Result: `S:9455` | | 100 SSQ3 | Result: `S:338350` |

Example 3 – generate a list using a DO…UNTIL…END construct
Type in the following program

```
« → xs xe dx « xe xs – dx / ABS 1. + xs → n x « xs DO
'x+dx' EVAL DUP 'x' STO UNTIL 'x≥xe' END n →LIST » » »
```

and store it in variable GLIS3.

- Check out that the program call 0.5 ENTER 2.5 ENTER 0.5 ENTER GLIS3 produces the list {0.5 1. 1.5 2. 2.5}.
- To see step-by-step operation use the program DBUG for a short list, for example:

| | |
|---|---|
| VAR 1 SPC 1.5 SPC 0.5 ENTER | Enter parameters 1 1.5 0.5 |
| [‘] GLIS3 ENTER | Enter the program name in level 1 |
| ← PRG NXT NXT RUN DBG | Start the debugger. |

Use SST↓ to step into the program and see the detailed operation of each command.

## The WHILE construct

The general structure of this command is:

```
WHILE logical_statement REPEAT program_statements END
```

The WHILE statement will repeat the `program_statements` while `logical_statement` is true (non zero). If not, program control is passed to the statement right after END. The `program_statements` must include a loop index that gets modified before the `logical_statement` is checked at the beginning of the next repetition. Unlike the DO command, if the first evaluation of logical_statement is false, the loop is never executed.

Example 1 – calculate the summation S using a WHILE…REPEAT…END construct
The following program calculates the summation

$$S = \sum_{k=0}^{n} k^2$$

Using a WHILE…REPEAT…END loop:

```
« 0. →n S « WHILE 'n≥0' REPEAT n SQ S + 'S' STO n 1 –
'n' STO END S "S" →TAG » »
```

Store this program in a variable SSUM4. Verify the following exercises: VAR

| | | | |
|---|---|---|---|
| 3 SSUM4 | Result: S:14 | 4 SSUM4 | Result: S:30 |
| 5 SSUM4 | Result: S:55 | 8 SSUM4 | Result: S:204 |
| 10 SSUM4 | Result: S:385 | 20 SSUM4 | Result: S:2870 |
| 30 SSUM4 | Result: S:9455 | 100 SSUM4 | Result: S:338350 |

Example 2 – generate a list using a WHILE…REPEAT…END construct
Type in the following program

```
« → xs xe dx « xe xs – dx / ABS 1. + xs → n x « xs
WHILE 'x<xe' REPEAT 'x+dx' EVAL DUP 'x' STO END n →LIST » »
»
```

and store it in variable GLIS4.

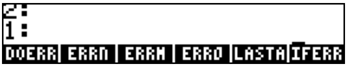
- Check out that the program call 0.5 ENTER 2.5 ENTER 0.5 ENTER GLIS4 produces the list {0.5 1. 1.5 2. 2.5}.
- To see step-by-step operation use the program DBUG for a short list, for example:

| | |
|---|---|
| VAR 1 SPC 1.5 SPC 0.5 ENTER | Enter parameters 1 1.5 0.5 |
| [‘] GLOSY ENTER | Enter the program name in level 1 |
| ← PRG NXT NXT RUN DBUG | Start the debugger. |

Use SST↓ to step into the program and see the detailed operation of each command.

## Errors and error trapping

The functions of the PRG/ERROR sub-menu provide ways to manipulate errors in the calculator, and trap errors in programs. The PRG/ERROR sub-menu, available through ← PRG NXT NXT ERROR , contains the following functions and sub-menus:



### DOERR

This function executes an user-define error, thus causing the calculator to behave as if that particular error has occurred. The function can take as argument either an integer number, a binary integer number, an error message, or the number zero (0). For example, in RPN mode, entering 5 ENTER DOERR, produces the following error message: *Error: Memory Clear*

If you enter #11h ENTER DOERR, produces the following message: *Error: Undefined FPTR Name*

If you enter “TRY AGAIN” ENTER DOERR, produces the following message: *TRY AGAIN*

Finally, 0 ENTER DOERR, produces the message: *Interrupted*

### ERRN

This function returns a number representing the most recent error. For example, if you try 0 1/x ON ERRN, you get the number #305h. This is the binary integer representing the error: *Infinite Result*

## ERRM

This function returns a character string representing the error message of the most recent error. For example, if you try 0 1/x ON ERRM, you get the following string: *"Infinite Result"*

## ERR0

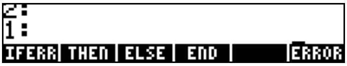
This function clears the last error number, so that, executing ERRN afterwards, will return # 0h. For example, if you try 0 1/x ON ERR0 ERRN, you get # 0h. Also, if you try 0 1/x ON ERR0 ERRM, you get the empty string " ".

## LASTARG

This function returns copies of the arguments of the command or function executed most recently. For example, in RPN mode, if you use: 3 ÷ 2 ENTER, and then use function LASTARG (LASTA), you will get the values 3 and 2 listed in the stack. Another example, in RPN mode, is the following: 5 TAN ENTER. Using LASTARG after these entries produces a 5.

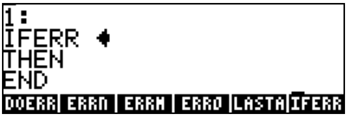
## Sub-menu IFERR

The IFERR sub-menu provides the following functions:

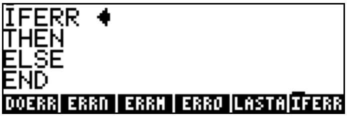


```
2:
1:
IFERR THEN ELSE END       ERROR
```

These are the components of the IFERR … THEN … END construct or of the IFERR … THEN … ELSE … END construct. Both logical constructs are used for trapping errors during program execution. Within the ERROR sub-menu, entering ↰ IFERR, or ↱ IFERR, will place the IFERR structure components in the stack, ready for the user to fill the missing terms, i.e.,



```
1:
IFERR ◆
THEN
END
DOERR ERRN ERRM ERR0 LASTA IFERR
```



```
IFERR ◆
THEN
ELSE
END
DOERR ERRN ERRM ERR0 LASTA IFERR
```

The general form of the two error-trapping constructs is as follows:

IF trap-clause THEN error-clause END

IF trap-clause THEN error-clause ELSE normal-clause END

The operation of these logical constructs is similar to that of the IF … THEN … END and of the IF … THEN … ELSE … END constructs. If an error is detected during the execution of the trap-clause, then the error-clause is executed. Otherwise, the normal-clause is executed.

As an example, consider the following program (ERR1) that takes as input two matrices, A and b, and checks if there is an error in the trap clause: A b / (RPN mode, i.e., A/b). If there is an error, then the program calls function LSQ (Least SQuares, see Chapter 11) to solve the system of equations:

```
« → A b « IFERR A b / THEN LSQ END »  »
```

Try it with the arguments A = [ [ 2, 3, 5 ] , [1, 2, 1 ] ] and b = [ [ 5 ] , [ 6 ] ]. A simple division of these two arguments produces an error: */Error: Invalid Dimension.*

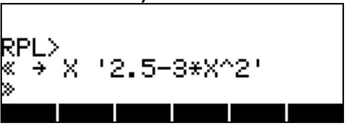
However, with the error-trapping construct of the program, ERR1, with the same arguments produces: [0.262295…, 0.442622…].

## User RPL programming in algebraic mode

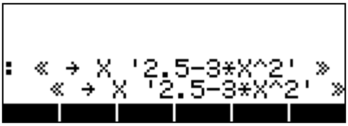
While all the programs presented earlier are produced and run in RPN mode, you can always type a program in User RPL when in algebraic mode by using function RPL>. This function is available through the command catalog. As an example, try creating the following program in algebraic mode, and store it into variable P2: `« → X ‘2.5-3*X^2’ »`

First, activate the RPL> function from the command catalog (↱ CAT ). All functions activated in ALG mode have a pair of parentheses attached to their name. The RPL> function is not exception, except that the parentheses must be removed before we type a program in the screen. Use the arrow keys (◄ ►) and the delete key (←) to eliminate the parentheses from the RPL>()

statement. At this point you will be ready to type the RPL program. The following figures show the RPL> command with the program before and after pressing the ENTER key.



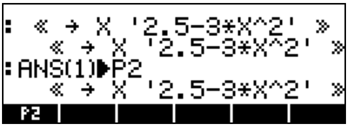
```
RPL>
« → X '2.5-3*X^2'
»
```



```
: « → X '2.5-3*X^2' »
   « → X '2.5-3*X^2' »
```

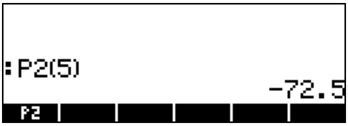
To store the program use the STO command as follows:

← ANS STO▶ ALPHA P 2 ENTER



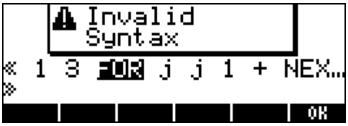
```
: « → X '2.5-3*X^2' »
   « → X '2.5-3*X^2' »
:ANS(1)▶P2
   « → X '2.5-3*X^2' »
```

An evaluation of program P2 for the argument X = 5 is shown in the next screen:



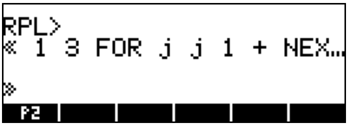
```
:P2(5)
                -72.5
```

While you can write programs in algebraic mode, without using the function RPL>, some of the RPL constructs will produce an error message when you press ENTER, for example:

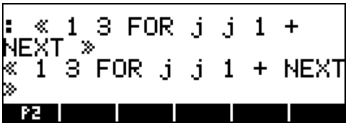


```
   Invalid
   Syntax
« 1 3 FOR j j 1 + NEX…
»
```

Whereas, using RPL, there is no problem when loading this program in algebraic mode:



```
RPL>
« 1 3 FOR j j 1 + NEX…
»
```



```
: « 1 3 FOR j j 1 +
NEXT »
« 1 3 FOR j j 1 + NEXT
»
```

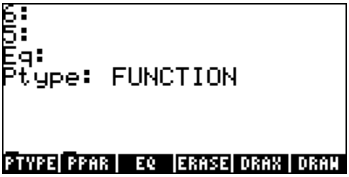
# Chapter 22
# Programs for graphics manipulation

This chapter includes a number of examples showing how to use the calculator's functions for manipulating graphics interactively or through the use of programs. As in Chapter 21 we recommend using RPN mode and setting system flag 117 to SOFT menu labels. « »

We introduce a variety of calculator graphic applications in Chapter 12. The examples of Chapter 12 represent interactive production of graphics using the calculator's pre-programmed input forms. It is also possible to use graphs in your programs, for example, to complement numerical results with graphics. To accomplish such tasks, we first introduce function in the PLOT menu.

## The PLOT menu

Commands for setting up and producing plots are available through the PLOT menu. You can access the PLOT menu by using: 8 1 . 0 1 ← PRG NXT MODES MENU MENU.



```
6:
5:
Eq:
Ptype: FUNCTION
```

The menu thus produced provides the user access to a variety of graphics functions. For application in subsequent examples, let's user-define the F3 (GRAPH) key to provide access to this menu as described below.

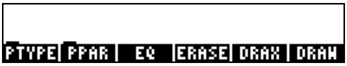
### User-defined key for the PLOT menu

Enter the following keystrokes to determine whether you have any user-defined keys already stored in your calculator: ← PRG NXT MODES KEYS RCLK. Unless you have user-defined some keys, you should get in return a list containing an S, i.e., {S}. This indicates that the Standard keyboard is the only key definition stored in your calculator.

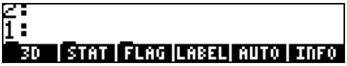
To user-define a key you need to add to this list a command or program followed by a reference to the key (see details in Chapter 20). Type the list `{ S << 81.01 MENU >> 13.0 }` in the stack and use function STOREKEYS (← PRG NXT MODES KEYS STOK) to user-define key F3 as the access to the PLOT menu. Verify that such list was stored in the calculator by using ← PRG NXT MODES KEYS RCLK.

**Note**: We will not work any exercise while presenting the PLOT menu, its functions or sub-menus. This section will be more like a tour of the contents of PLOT as they relate to the different type of graphs available in the calculator.

To activate a user defined key you need to press ← USER (same as the ALPHA key) before pressing the key or keystroke combination of interest. To activate the PLOT menu, with the key definition used above, press: ← USER F3. You will get the following menu (press NXT to move to second menu)



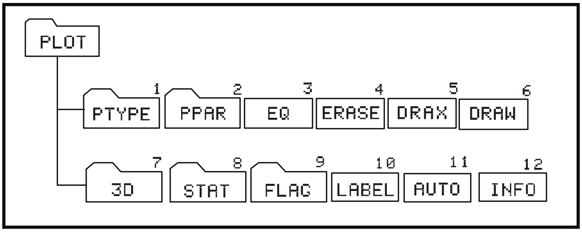
```
PTYPE PPAR  EQ  ERASE DRAX  DRAW
```



```
2:
1:
  3D   STAT  FLAG  LABEL AUTO  INFO
```

## Description of the PLOT menu

The following diagram shows the menus in PLOT. The number accompanying the different menus and functions in the diagram are used as reference in the subsequent description of those objects.



The soft menu key labeled 3D, STAT, FLAG, PTYPE, and PPAR, produce additional menus, which will be presented in more detail later. At this point we describe the functions directly accessible through soft menu keys for menu number 81.02. These are:

LABEL (10)
The function LABEL is used to label the axes in a plot including the variable names and minimum and maximum values of the axes. The variable names are selected from information contained in the variable PPAR.

AUTO (11)
The function AUTO (AUTOscale) calculates a display range for the y-axis or for both the x- and y-axes in two-dimensional plots according to the type of plot defined in PPAR. For any of the three-dimensional graphs the function AUTO produces no action. For two-dimensional plots, the following actions are performed by AUTO:

- FUNCTION: based on the plotting range of x, it samples the function in EQ and determines the minimum and maximum values of y.
- CONIC: sets the y-axis scale equal to the x-axis scale
- POLAR: based on the values of the independent variable (typically θ), it samples the function in EQ and determines minimum and maximum values of both x and y.
- PARAMETRIC: produces a similar result as POLAR based on the values of the parameter defining the equations for x and y.
- TRUTH: produces no action.
- BAR: the x-axis range is set from 0 to *n+1* where *n* is the number of elements in ΣDAT. The range of values of y is based on the contents of ΣDAT. The minimum and maximum values of y are determined so that the x-axis is always included in the graph.
- HISTOGRAM: similar to BAR.
- SCATTER: sets x- and y-axis range based on the contents of the independent and dependent variables from ΣDAT.

INFO (12)
The function INFO is interactive only (i.e., it cannot be programmed). When the corresponding soft menu key is pressed it provides information about the current plot parameters.

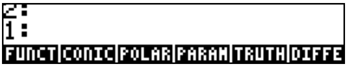
EQ (3)
The variable name EQ is reserved by the calculator to store the current equation in plots or solution to equations (see chapter …). The soft menu key labeled EQ in this menu can be used as it would be if you have your variable menu available, e.g., if you press [ EQ ] it will list the current contents of that variable.

ERASE (4)
The function ERASE erases the current contents of the graphics window. In programming, it can be used to ensure that the graphics window is cleared before plotting a new graph.

DRAX (5)
The function DRAX draws the axes in the current plot, if any is visible.

DRAW (6)
The function DRAW draws the plot defined in PPAR.

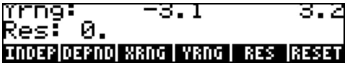
**The PTYPE menu under PLOT (1)**
The PTYPE menu lists the name of all two-dimensional plot types pre-programmed in the calculator. The menu contains the following menu keys:



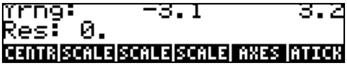
```
2:
1:
```

These keys correspond to the plot types *Function, Conic, Polar, Parametric, Truth*, and *Diff Eq*, presented earlier. Pressing one of these soft menu keys, while typing a program, will place the corresponding function call in the program. Press NXT PLOT to get back to the main PLOT menu.

**The PPAR menu (2)**
The PPAR menu lists the different options for the PPAR variable as given by the following soft menu key labels. Press NXT to move to next menus:



```
Yrng:      -3.1        3.2
Res: 0.
```

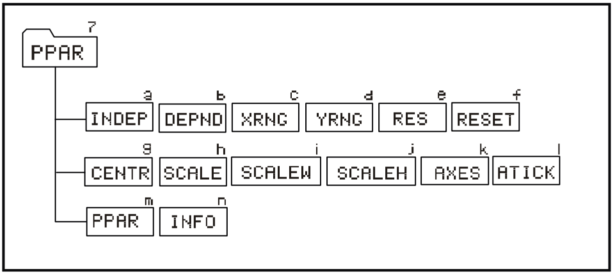


```
Yrng:      -3.1        3.2
Res: 0.
```

```
Yrng:      -3.1       3.2
Res: 0.
PPAR INFO              PLOT
```

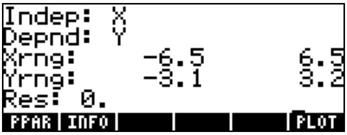
**Note**: the SCALE commands shown here actually represent SCALE, SCALEW, SCALEH, in that order.

The following diagram illustrates the functions available in the PPAR menu. The letters attached to each function in the diagram are used for reference purposes in the description of the functions shown below.



INFO (n) and PPAR (m)

If you press INFO, or enter ↱ PPAR, while in this menu, you will get a listing of the current PPAR settings, for example:



```
Indep: X
Depnd: Y
Xrng:      -6.5       6.5
Yrng:      -3.1       3.2
Res: 0.
PPAR INFO              PLOT
```

This information indicates that X is the independent variable (Indep), Y is the dependent variable (Depnd), the x-axis range goes from –6.5 to 6.5 (Xrng), the y-axis range goes from –3.1 to 3.2 (Yrng). The last piece of information in the screen, the value of Res (resolution) determines the interval of the independent variable used for generating the plot.

The soft menu key labels included in the PPAR(2) menu represent commands that can be used in programs. These commands include:

INDEP (a)

The command INDEP specifies the independent variable and its plotting range. These specifications are stored as the third parameter in the variable PPAR. The default value is 'X'. The values that can be assigned to the independent variable specification are:

- A variable name, e.g., 'Vel'
- A variable name in a list, e.g., { Vel }
- A variable name and a range in a list, e.g., { Vel 0 20 }
- A range without a variable name, e.g., { 0 20 }
- Two values representing a range, e.g., 0 20

In a program, any of these specifications will be followed by the command INDEP.

DEPND (b)

The command DEPND specifies the name of the dependent variable. For the case of TRUTH plots it also specifies the plotting range. The default value is Y. The type of specifications for the DEPND variable are the same as those for the INDEP variable.

XRNG (c) and YRNG (d)

The command XRNG specifies the plotting range for the x-axis, while the command YRNG specifies the plotting range for the y-axis. The input for any of these commands is two numbers representing the minimum and maximum values of x or y. The values of the x- and y-axis ranges are stored as the ordered pairs ($x_{min}$, $y_{min}$) and ($x_{max}$, $y_{max}$) in the two first elements of the variable PPAR. Default values for $x_{min}$ and $x_{max}$ are -6.5 and 6.5, respectively. Default values for $x_{min}$ and $x_{max}$ are –3.1 and 3.2, respectively.

RES (e)

The RES (RESolution) command specifies the interval between values of the independent variable when producing a specific plot. The resolution can be expressed in terms of user units as a real number, or in terms of pixels as a binary integer (numbers starting with #, e.g., #10). The resolution is stored as the fourth item in the PPAR variable.

CENTR (g)
The command CENTR takes as argument an ordered pair (x,y) or a value x, and adjusts the first two elements in the variable PPAR, i.e., ($x_{min}$, $y_{min}$) and ($x_{max}$, $y_{max}$), so that the center of the plot is (x,y) or (x,0), respectively.

SCALE (h)
The SCALE command determines the plotting scale represented by the number of user units per tick mark. The default scale is 1 user-unit per tick mark. When the command SCALE is used, it takes as arguments two numbers, $x_{scale}$ and $y_{scale}$, representing the new horizontal and vertical scales. The effect of the SCALE command is to adjust the parameters ($x_{min}$, $y_{min}$) and ($x_{max}$, $y_{max}$) in PPAR to accommodate the desired scale. The center of the plot is preserved.

SCALEW (i)
Given a factor $x_{factor}$, the command SCALEW multiplies the horizontal scale by that factor. The W in SCALEW stands for 'width.' The execution of SCALEW changes the values of $x_{min}$ and $x_{max}$ in PPAR.

SCALEH (j)
Given a factor $y_{factor}$, the command SCALEH multiplies the vertical scale by that factor. The H in SCALEH stands for 'height.' The execution of SCALEW changes the values of $y_{min}$ and $y_{max}$ in PPAR.

**Note**: Changes introduced by using SCALE, SCALEW, or SCALEH, can be used to zoom in or zoom out in a plot.

ATICK (l)
The command ATICK (Axes TICK mark) is used to set the tick-mark annotations for the axes. The input value for the ATICK command can be one of the following:

- A real value x : sets both the x- and y-axis tick annotations to x units
- A list of two real values { x y }: sets the tick annotations in the x- and y-axes to x and y units, respectively.
- A binary integer #n: sets both the x- and y-axis tick annotations to #n pixels

A list of two binary integers {#n #m}: sets the tick annotations in the x- and y-axes to #n and #m pixels, respectively.

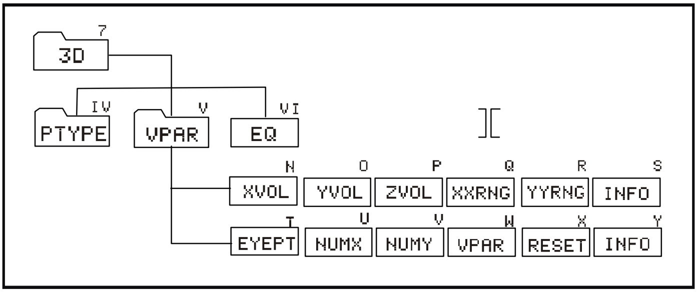
AXES (k)
The input value for the axes command consists of either an ordered pair (x,y) or a list {(x,y) *atick* "x-axis label" "y-axis label"}. The parameter *atick* stands for the specification of the tick marking annotations as described above for the command ATICK. The ordered pair represents the center of the plot. If only an ordered pair is given as input to AXES, only the axes origin is altered. The argument to the command AXES, whether an ordered pair or a list of values, is stored as the fifth parameter in PPAR.

To return to the PLOT menu, press PLOT.

Press NXT to reach the second menu of the PLOT menu set.

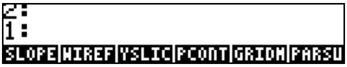
RESET (f)
This button will reset the plot parameters to default values.

**The 3D menu within PLOT (7)**
The 3D menu contains two sub-menus, PTYPE and VPAR, and one variable, EQ. We are familiar already with the meaning of EQ, therefore, we will concentrate on the contents of the PTYPE and VPAR menus. The diagram below shows the branching of the 3D menu.



### The PTYPE menu within 3D (IV)

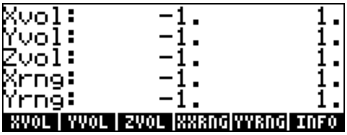
The PTYPE menu under 3D contains the following functions:



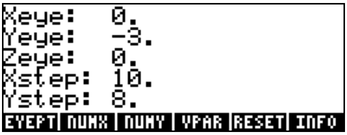
These functions correspond to the graphics options *Slopefield, Wireframe, Y-Slice, Ps-Contour, Gridmap* and *Pr-Surface* presented earlier in this chapter. Pressing one of these soft menu keys, while typing a program, will place the corresponding function call in the program. Press NXT 3D to get back to the main 3D menu.

### The VPAR menu within 3D (V)

The variable VPAR stands for Volume PARameters, referring to a parallelepiped in space within which the three-dimensional graph of interest is constructed. When press [VPAR] in the 3D menu, you will get the following functions. Press NXT to move to the next menu:



```
Xvol:      -1.        1.
Yvol:      -1.        1.
Zvol:      -1.        1.
Xrng:      -1.        1.
Yrng:      -1.        1.
```



```
Xeye:   0.
Yeye:   -3.
Zeye:   0.
Xstep:  10.
Ystep:  8.
```

Next, we describe the meaning of these functions:

INFO (S) and VPAR (W)

When you press INFO (S) you get the information shown in the left-hand side screen shot above. The ranges in *Xvol*, *Yvol*, and *Zvol* describe the extent of the parallelepiped in space where the graph will be generated. *Xrng* and *Yrng* describe the range of values of x and y, respectively, as independent variables in the x-y plane that will be used to generate functions of the form z = f(x,y).

Press NXT and INFO (Y) to obtain the information in the right-hand side screen shot above. These are the value of the location of the viewpoint for the three-dimensional graph (Xeye, Yeye, Zeye), and of the number of steps in x and y to generate a grid for surface plots.

XVOL (N), YVOL (O), and ZVOL (P)
These functions take as input a minimum and maximum value and are used to specify the extent of the parallelepiped where the graph will be generated (the viewing parallelepiped). These values are stored in the variable VPAR. The default values for the ranges XVOL, YVOL, and ZVOL are –1 to 1.

XXRNG (Q) and YYRNG (R)
These functions take as input a minimum and maximum value and are used to specify the ranges of the variables x and y to generate functions z = f(x,y). The default value of the ranges XXRNG and YYRNG will be the same as those of XVOL and YVOL.

EYEPT (T)
The function EYEPT takes as input real values x, y, and z representing the location of the viewpoint for a three-dimensional graph. The viewpoint is a point in space from which the three-dimensional graph is observed. Changing the viewpoint will produce different views of the graph. The figure below illustrates the idea of the viewpoint with respect to the actual graphic space and its projection in the plane of the screen.

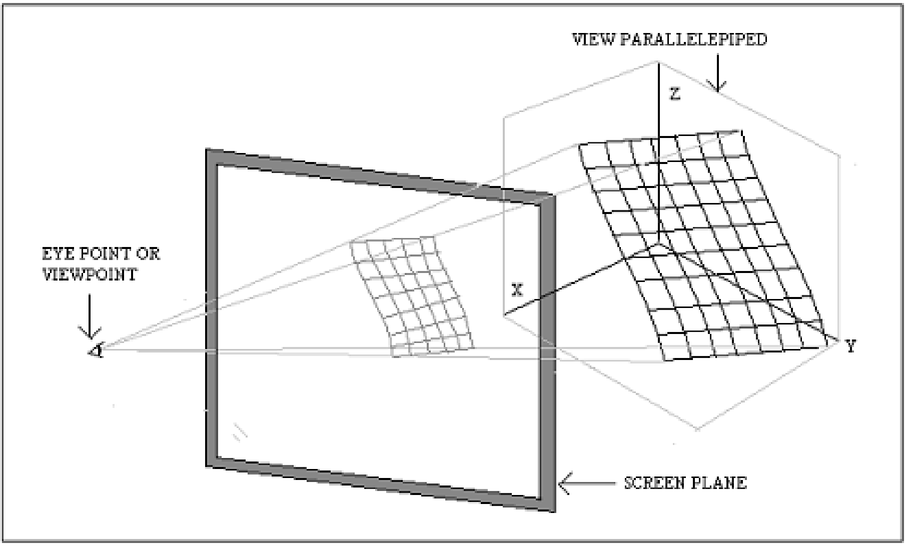
NUMX(U) and NUMY (V)
The functions NUMX and NUMY are used to specify the number of points or steps along each direction to be used in the generation of the base grid from which to obtain values of z = f(x,y).

VPAR (W)
This is just a reference to the variable VPAR.

RESET (X)
Resets parameters in screen to their default values.

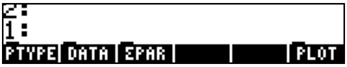
Press NXT 3D to return to the 3D menu.

Press PLOT to return to the PLOT menu.

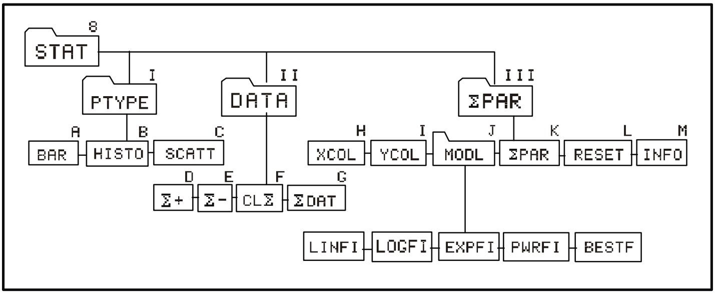


## The STAT menu within PLOT

The STAT menu provides access to plots related to statistical analysis. Within this menu we find the following menus:

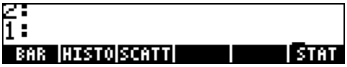


The diagram below shows the branching of the STAT menu within PLOT. The numbers and letters accompanying each function or menu are used for reference in the descriptions that follow the figure.



### *The PTYPE menu within STAT (I)*

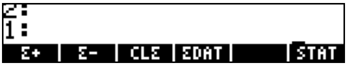
The PTYPE menu provides the following functions:



These keys correspond to the plot types *Bar (A), Histogram (B),* and *Scatter(C),* presented earlier. Pressing one of these soft menu keys, while typing a program, will place the corresponding function call in the program. Press STAT to get back to the STAT menu.

### *The DATA menu within STAT (II)*

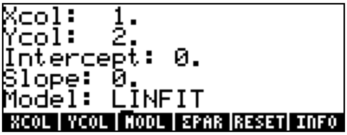
The DATA menu provides the following functions:



The functions listed in this menu are used to manipulate the ΣDAT statistical matrix. The functions Σ+ (D) and Σ- (E), add or remove data rows from the matrix ΣDAT. CLΣ (F) clears the ΣDAT (G) matrix, and the soft menu key labeled ΣDAT is just used as a reference for interactive applications. More details on the use of these functions are presented in a later chapter on statistical applications. Press STAT to return to the STAT menu.

### *The ΣPAR menu within STAT (III)*

The ΣPAR menu provides the following functions:



```
Xcol:  1.
Ycol:  2.
Intercept: 0.
Slope: 0.
Model: LINFIT
```

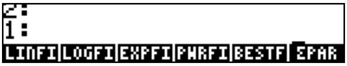
INFO (M) and ΣPAR (K)

The key INFO in ΣPAR provides the information shown in the screen shot above. The information listed in the screen is contained in the variable ΣPAR. The values shown are the default values for the x-column, y-column, intercept and slope of a data fitting model, and the type of model to be fit to the data in ΣDAT.

XCOL (H)
The command XCOL is used to indicate which of the columns of ΣDAT, if more than one, will be the x- column or independent variable column.

YCOL (I)
The command YCOL is used to indicate which of the columns of ΣDAT, if more than one, will be the y- column or dependent variable column.

MODL (J)
The command MODL  refers to the model to be selected to fit the data in SDAT, if a data fitting is implemented.  To see which options are available, press MODL.  You will get the following menu:



These functions correspond to Linear Fit, Logarithmic Fit, Exponential Fit, Power Fit, or Best Fit.  Data fitting is described in more detail in a later chapter.   Press ΣPAR to return to the ΣPAR menu.

ΣPAR (K)
ΣPAR is just a reference to the variable SPAR for interactive use.

RESET (L)
This function resets the contents of ΣPAR to its default values.

Press NXT STAT to return to the STAT menu.  Press [PLOT] to return to the main PLOT  menu.

**The FLAG menu within PLOT**
The FLAG  menu is actually interactive, so that you can select any of the following options:

- AXES: when selected, axes are shown if visible within the plot area or volume.
- CNCT: when selected the plot is produced so that individual points are connected.
- SIMU: when selected, and if more than one graph is to be plotted in the same set of axes, plots all the graphs simultaneously.

Press PLOT to return to the PLOT menu.

## Generating plots with programs

Depending on whether we are dealing with a two-dimensional graph defined by a function, by data from ΣDAT, or by a three-dimensional function, you need to set up the variables PPAR, ΣPAR, and /or VPAR before generating a plot in a program. The commands shown in the previous section help you in setting up such variables.

Following we describe the general format for the variables necessary to produce the different types of plots available in the calculator.

### Two-dimensional graphics

The two-dimensional graphics generated by functions, namely, *Function, Conic, Parametric, Polar, Truth and Differential Equation*, use PPAR with the format:

{ ($x_{min}$, $y_{min}$) ($x_{max}$, $y_{max}$) indep res axes ptype depend }

The two-dimensional graphics generated from data in the statistical matrix ΣDAT, namely, *Bar, Histogram,* and *Scatter*, use the ΣPAR variable with the following format:

```
{x-column y-column slope intercept model}
```

while at the same time using PPAR with the format shown above.

The meaning of the different parameters in PPAR and ΣPAR were presented in the previous section.

## Three-dimensional graphics

The three-dimensional graphics available, namely, options *Slopefield, Wireframe, Y-Slice, Ps-Contour, Gridmap* and *Pr-Surface*, use the VPAR variable with the following format:

$\{x_{left},\ x_{right},\ y_{near},\ y_{far},\ z_{low},\ z_{high},\ x_{min},\ x_{max},\ y_{min},\ y_{max},\ x_{eye},\ y_{eye},\ z_{eye},\ x_{step},\ y_{step}\}$

These pairs of values of x, y, and z, represent the following:

- Dimensions of the view parallelepiped ($x_{left},\ x_{right},\ y_{near},\ y_{far},\ z_{low},\ z_{high}$)
- Range of x and y independent variables ($x_{min},\ x_{max},\ y_{min},\ y_{max}$)
- Location of viewpoint ($x_{eye},\ y_{eye},\ z_{eye}$)
- Number of steps in the x- and y-directions ($x_{step},\ y_{step}$)

Three-dimensional graphics also require the PPAR variable with the parameters shown above.

## The variable EQ

All plots, except those based on ΣDAT, also require that you define the function or functions to be plotted by storing the expressions or references to those functions in the variable EQ.

In summary, to produce a plot in a program you need to load EQ, if required. Then load PPAR, PPAR and SPAR, or PPAR and VPAR. Finally, use the name of the proper plot type: FUNCTION, CONIC, POLAR, PARAMETRIC, TRUTH, DIFFEQ, BAR, HISTOGRAM, SCATTER, SLOPE, WIREFRAME, YSLICE, PCONTOUR, GRIDMAP, or PARSURFACE, to produce your plot.

## Examples of interactive plots using the PLOT menu

To better understand the way a program works with the PLOT commands and variables, try the following examples of interactive plots using the PLOT menu.

Example 1 – <u>A function plot:</u>

| | |
|---|---|
| ← USER F3 | Get PLOT menu (*) |
| PTYPE FUNCT | Select FUNCTION as the plot type |
| ‘√r’ ENTER ← EQ | Store function ‘√r’ into EQ |

| | |
|---|---|
| PPAR | Show plot parameters |
| ALPHA ← R ENTER INDEP | Define 'r' as the indep. variable |
| ALPHA ← S ENTER DEPND | Define 's' as the dependent variable |
| 1 +/- SPC 10 XRNG | Define (-1, 10) as the x-range |
| 1 +/- SPC 5 YRNG NXT | Define (-1, 5) as the y-range |
| { (0,0) {.4 .2} "Rs" "Sr"} | Axes definition list |
| AXES | Define axes center, ticks, labels |
| NXT PLOT | Return to PLOT menu |
| ERASE DRAX NXT LABEL | Erase picture, draw axes, labels |
| NXT DRAW | Draw function and show picture |
| EDIT NXT MENU | Removes menu labels |
| NXT NXT PICT CANCL | Returns to normal calculator display |

(*) PLOT menu available through user-defined key F3 as shown earlier in this Chapter.

Example 2 – <u>A parametric plot</u>:

| | |
|---|---|
| ← USER F3 | Get PLOT menu |
| TYPE PARA | Select PARAMETRIC as the plot type |
| { 'SIN(t)+i*SIN(2*t)' } ENTER | Define complex function X+iY |
| ← EQ | Store complex function into EQ |
| PPAR | Show plot parameters |
| {t 0 6.29} ENTER INDEP | Define 't' as the indep.variable |
| ALPHA Y ENTER DEPND | Define 'Y' as the dependent variable |
| 2.2 +/- SPC 2.2 XRNG | Define (-2.2,2.2) as the x-range |
| 1.1 +/- SPC 1.1 YRNG NXT | Define (-1.1,1.1) as the y-range |
| { (0,0) {} {.4 .2} "X(t)" "Y(t)"} ENTER | Axes definition list |
| AXES | Define axes center, ticks, labels |
| NXT PLOT | Return to PLOT menu |
| ERASE DRAX NXT LABEL | Erase picture, draw axes, labels |
| NXT DRAW | Draw function and show picture |
| EDIT NXT MENU NXT NXT PICT CANCL | Finish plot |

Example 3 – <u>A polar plot</u>:

| | |
|---|---|
| ← USER F3 | Get PLOT menu |
| TYPE POLAR | Select POLAR as the plot type |

| | |
|---|---|
| '1+SIN(θ)' ENTER ← EQ | Store complex funct. r = f(θ) into EQ |
| PPAR | Show plot parameters |
| { θ 0 6.29} ENTER INDEP | Define 'θ' as the indep. Variable |
| ALPHA Y ENTER DEPND | Define 'Y' as the dependent variable |
| 3 +/- SPC 3 XRNG | Define (-3,3) as the x-range |
| 0.5 +/- SPC 2.5 YRNG NXT | Define (-0.5,2.5) as the y-range |
| { (0,0) {.5 .5} "x" "y"} ENTER | Axes definition list |
| AXES | Define axes center, ticks, labels |
| NXT PLOT | Return to PLOT menu |
| ERASE DRAX NXT LABEL | Erase picture, draw axes, labels |
| NXT DRAW | Draw function and show picture |
| EDIT NXT MENU | Remove menu labels |
| NXT NXT PICT CANCL | Return to normal calculator display |

From these examples we see a pattern for the interactive generation of a two-dimensional graph through the PLOT menu:

1 – Select PTYPE.
2 – Store function to plot in variable EQ (using the proper format, e.g., 'X(t)+iY(t)' for PARAMETRIC).
3 – Enter name (and range, if necessary) of independent and dependent variables
4 – Enter axes specifications as a list { center atick x-label y-label }
5 – Use ERASE, DRAX, LABEL, DRAW to produce a fully labeled graph with axes

This same approach can be used to produce plots with a program, except that in a program you need to add the command PICTURE after the DRAW function is called to recall the graphics screen to the stack.

## Examples of program-generated plots

In this section we show how to implement with programs the generation of the last three examples. Activate the PLOT menu before you start typing the program to facilitate entering graphing commands (← USER F3 , see above).

Example 1 – A function plot. Enter the following program:

```
«                                  Start program
{PPAR EQ} PURGE                    Purge current PPAR and EQ
'√r' STEQ                          Store '√r' into EQ
'r' INDEP                          Set independent variable to 'r'
's' DEPND                          Set dependent variable to 's'
FUNCTION                           Select FUNCTION as the plot type
{ (0.,0.) {.4 .2}
"Rs" "Sr" } AXES                   Set axes information
-1. 5. XRNG                        Set x range
-1. 5. YRNG                        Set y range
ERASE DRAW DRAX LABEL              Erase & draw plot, axes, and labels
PICTURE »                          Recall graphics screen to stack
```

Store the program in variable PLOT1. To run it, press VAR, if needed, then press PLOT1.

Example 2 – A parametric plot. Enter the following program:

```
«                                  Start program
RAD {PPAR EQ} PURGE                Change to radians, purge vars.
'SIN(t)+i*SIN(2*t)' STEQ           Store 'X(t)+iY(t)' into EQ
{ t 0. 6.29} INDEP                 Set indep. variable to 'r', with range
'Y' DEPND                          Set dependent variable to 'Y'
PARAMETRIC                         Select PARAMETRIC as the plot type
{ (0.,0.) {.5 .5} "X(t)"
"Y(t)" } AXES                      Set axes information
-2.2 2.2 XRNG                      Set x range
-1.1 1.1 YRNG                      Set y range
ERASE DRAW DRAX LABEL              Erase & draw plot, axes, and labels
PICTURE                            Recall graphics screen to stack
»                                  End program
```

Store the program in variable PLOT2. To run it, press VAR, if needed, then press PLOT2.

Example 3 – A polar plot. Enter the following program:

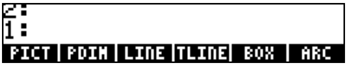
```
«                                   Start program
RAD {PPAR EQ} PURGE                 Change to radians, purge vars.
'1+SIN(θ)' STEQ                     Store 'f(θ)' into EQ
{ θ 0. 6.29} INDEP                  Set indep. variable to 'θ', with
range
'Y' DEPND                           Set dependent variable to 'Y'
POLAR                               Select POLAR as the plot type
{ (0.,0.) {.5 .5}
"x" "y"} AXES                       Set axes information
-3.  3. XRNG                        Set x range
-.5  2.5 YRNG                       Set y range
ERASE DRAW DRAX LABEL               Erase & draw plot, axes, and labels
PICTURE                             Recall graphics screen to stack
»                                   End program
```

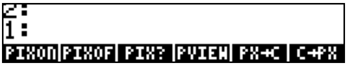
Store the program in variable PLOT3. To run it, press VAR, if needed, then press PLOT3.

These exercises illustrate the use of PLOT commands in programs. They just scratch the surface of programming applications of plots. I invite the reader to try their own exercises on programming plots.

## Drawing commands for use in programming

You can draw figures in the graphics window directly from a program by using commands such as those contained in the PICT menu, accessible by ← PRG NXT PICT. The functions available in this menu are the following. Press NXT to move to next menu:





Obviously, the commands LINE, TLINE, and BOX, perform the same operations as their interactive counterpart, given the appropriate input. These and the other functions in the PICT menu refer to the graphics windows whose x- and y-ranges are determined in the variable PPAR, as demonstrated above for different graph types. The functions in the PICT command are described next:

## PICT

This soft key refers to a variable called PICT that stores the current contents of the graphics window. This variable name, however, cannot be placed within quotes, and can only store graphics objects. In that sense, PICT is like no other calculator variables.

## PDIM

The function PDIM takes as input either two ordered pairs ($x_{min}$,$y_{min}$) ($x_{max}$ $y_{max}$) or two binary integers #w and #h. The effect of PDIM is to replace the current contents of PICT with an empty screen. When the argument is ($x_{min}$,$y_{min}$) ($x_{max}$ $y_{max}$), these values become the range of the user-defined coordinates in PPAR. When the argument is #w and #h, the ranges of the user-defined coordinates in PPAR remain unchanged, but the size of the graph changes to #h × #v pixels.

### PICT and the graphics screen

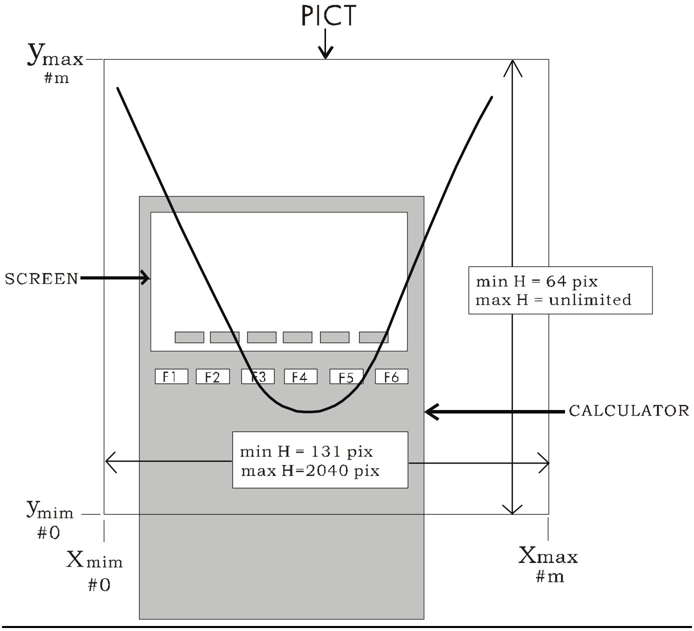
PICT, the storage area for the current graph, can be thought of as a two dimensional graph with a minimum size of 131 pixels wide by 64 pixels high. The maximum width of PICT is 2048 pixels, with no restriction on the maximum height. A pixel is each one of the dots in the calculator's screen that can be turned on (dark) or off (clear) to produce text or graphs. The calculator screen has 131 pixels by 64 pixels, i.e., the minimum size for PICT. If your PICT is larger than the screen, then the PICT graph can be thought of as a two dimensional domain that can be scrolled through the calculator's screen, as illustrated in the diagram shown next.

## LINE

This command takes as input two ordered pairs ($x_1$,$y_1$) ($x_2$, $y_2$), or two pairs of pixel coordinates {#$n_1$ #$m_1$} {#$n_2$ #$m_2$}. It draws the line between those coordinates.

## TLINE

This command (Toggle LINE) takes as input two ordered pairs ($x_1$,$y_1$) ($x_2$, $y_2$), or two pairs of pixel coordinates {#$n_1$ #$m_1$} {#$n_2$ #$m_2$}. It draws the line between those coordinates, turning off pixels that are on in the line path and vice versa.



## BOX

This command takes as input two ordered pairs $(x_1,y_1)$ $(x_2, y_2)$, or two pairs of pixel coordinates $\{\#n_1\ \#m_1\}$ $\{\#n_2\ \#m_2\}$. It draws the box whose diagonals are represented by the two pairs of coordinates in the input.

## ARC

This command is used to draw an arc. ARC takes as input the following objects:

- Coordinates of the center of the arc as (x,y) in user coordinates or {#n, #m} in pixels.
- Radius of arc as r (user coordinates) or #k (pixels).
- Initial angle $\theta_1$ and final angle $\theta_2$.

## PIX?, PIXON, and PIXOFF

These functions take as input the coordinates of point in user coordinates, (x,y), or in pixels {#n, #m}.

- PIX? Checks if pixel at location (x,y) or {#n, #m} is on.
- PIXOFF turns off pixel at location (x,y) or {#n, #m}.
- PIXON turns on pixel at location (x,y) or {#n, #m}.

## PVIEW

This command takes as input the coordinates of a point as user coordinates (x,y) or pixels {#n, #m}, and places the contents of PICT with the upper left corner at the location of the point specified. You can also use an empty list as argument, in which case the picture is centered in the screen. PVIEW does not activate the graphics cursor or the picture menu. To activate any of those features use PICTURE.

## PX→C

The function PX→C converts pixel coordinates {#n #m} to user-unit coordinates (x,y).

## C→PX

The function C→PX converts user-unit coordinates (x,y) to pixel coordinates {#n #m}.

## Programming examples using drawing functions

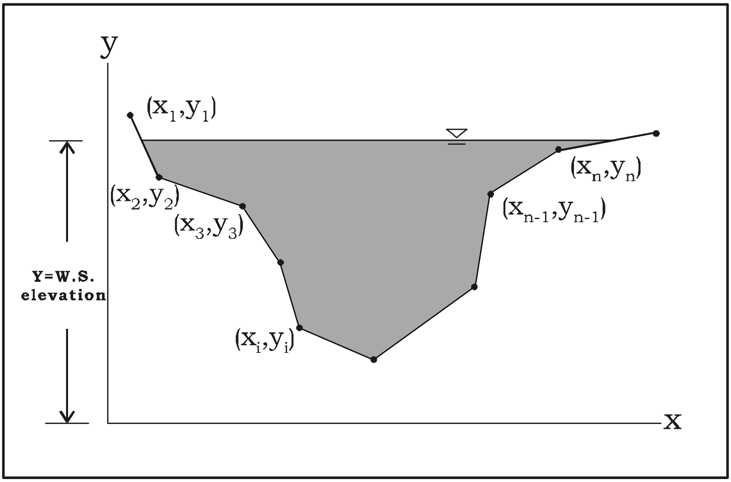
In this section we use the commands described above to produce graphics with programs. Program listing are provided in the attached diskette or CD ROM.

Example 1 - A program that uses drawing commands
The following program produces a drawing in the graphics screen. (This program has no other purpose than to show how to use calculator commands to produce drawings in the display.)

```
«                                    Start program
DEG                                  Select degrees for angular measures
0. 100. XRNG                         Set x range
0. 50.  YRNG                         Set y range
ERASE                                Erase picture
(5., 2.5) (95., 47.5) BOX            Draw box from (5,5) to (95,95)
(50., 50.) 10. 0. 360. ARC           Draw a circle center (50,50), r =10.
(50., 50.) 12. –180. 180. ARC        Draw a circle center (50,50), r= 12.
1 8 FOR j                            Draw 8 lines within the circle
    (50., 50.) DUP                   Lines are centered as (50,50)
    '12*COS(45*(j-1))' →NUM          Calculate x, other end at 50 + x
    '12*SIN(45*(j-1))' →NUM          Calculates y, other end at 50 + y
     R → C                           Convert x y to (x,y), complex num.
     +                               Add (50,50) to (x,y)
     LINE                            Draw the line
NEXT                                 End of FOR loop
{ } PVIEW                            Show picture
»
```

<u>Example 2</u> - A program to plot a natural river cross-section
This application may be useful for determining area and wetted perimeters of natural river cross-sections. Typically, a natural river cross section is surveyed and a series of points, representing coordinates x and y with respect to an arbitrary set of coordinates axes. These points can be plotted and a sketch of the cross section produced for a given water surface elevation. The figure below illustrate the terms presented in this paragraph.

The program, available in the diskette or CD ROM that comes with your calculator, utilizes four sub-programs FRAME, DXBED, GTIFS, and INTRP. The main program, called XSECT, takes as input a matrix of values of x and y, and the elevation of the water surface Y (see figure above), in that order. The program produces a graph of the cross section indicating the input data with points in the graph, and shows the free surface in the cross-section.



It is suggested that you create a separate sub-directory to store the programs. You could call the sub-directory RIVER, since we are dealing with irregular open channel cross-sections, typical of rivers.

To see the program XSECT in action, use the following data sets. Enter them as matrices of two columns, the first column being x and the second one y. Store the matrices in variables with names such as XYD1 (X-Y Data set 1) and XYD2 (X-Y Data set 2). To run the program place one of the data sets in the stack, e.g., VAR XYD1, then type in a water surface elevation, say 4.0, and press XSECT. The calculator will show an sketch of the cross-section with the corresponding water surface. To exit the graph display, press ON.

Try the following examples:

XYD1 2 XSECT
XYD1 3 XSECT
XYD1 4 XSECT
XYD1 6 XSECT

Please be patient when running program XSECT. Due to the relatively large number of graphics functions used, not counting the numerical iterations, it may take some time to produce the graph (about 1 minute).

**Data set 1**

| x | y |
|---|---|
| 0.4 | 6.3 |
| 1.0 | 4.9 |
| 2.0 | 4.3 |
| 3.4 | 3.0 |
| 4.0 | 1.2 |
| 5.8 | 2.0 |
| 7.2 | 3.8 |
| 7.8 | 5.3 |
| 9.0 | 7.2 |

**Data set 2**

| x | y |
|---|---|
| 0.7 | 4.8 |
| 1.0 | 3.0 |
| 1.5 | 2.0 |
| 2.2 | 0.9 |
| 3.5 | 0.4 |
| 4.5 | 1.0 |
| 5.0 | 2.0 |
| 6.0 | 2.5 |
| 7.1 | 2.0 |
| 8.0 | 0.7 |
| 9.0 | 0.0 |
| 10.0 | 1.5 |
| 10.5 | 3.4 |
| 11.0 | 5.0 |

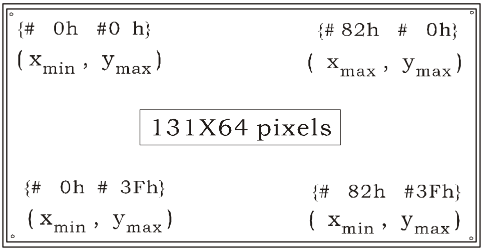
**Note**: The program FRAME, as originally programmed (see diskette or CD ROM), does not maintain the proper scaling of the graph. If you want to maintain proper scaling, replace FRAME with the following program:

```
« STOΣ MINΣ MAXΣ 2 COL→ DUP →COL DROP – AXL ABS AXL 20
/ DUP NEG SWAP 2 COL→ + →ROW DROP SWAP → yR xR « 131
DUP R→B SWAP yR OBJ→ DROP – xR OBJ→ DROP - / * FLOOR
R→B PDIM yR OBJ→ DROP YRNG xR OBJ→ DROP XRNG ERASE » »
```

This program keeps the width of the PICT variable at 131 pixels – the minimum pixel size for the horizontal axis – and adjusts the number of pixels in the vertical axes so that a 1:1 scale is maintained between the vertical and horizontal axes.

## Pixel coordinates

The figure below shows the graphic coordinates for the typical (minimum) screen of 131×64 pixels. Pixels coordinates are measured from the top left corner of the screen {# 0h # 0h}, which corresponds to user-defined coordinates ($x_{min}$, $y_{max}$). The maximum coordinates in terms of pixels correspond to the lower right corner of the screen {# 82h #3Fh}, which in user-coordinates is the point ($x_{max}$, $y_{min}$). The coordinates of the two other corners both in pixel as well as in user-defined coordinates are shown in the figure.

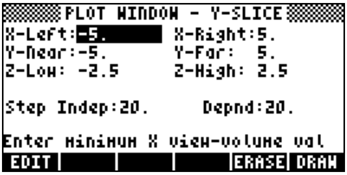

{# 0h #0 h}
( $x_{min}$ , $y_{max}$ )
{# 82h # 0h}
( $x_{max}$ , $y_{max}$ )
131X64 pixels
{# 0h # 3Fh}
( $x_{min}$ , $y_{max}$ )
{# 82h #3Fh}
( $x_{min}$ , $y_{max}$ )

## Animating graphics

Herein we present a way to produce animation by using the Y-Slice plot type. Suppose that you want to animate the traveling wave, f(X,Y) = 2.5 sin(X-Y). We can treat the X as time in the animation producing plots of f(X,Y) vs. Y for different values of X. To produce this graph use the following:

- ← 2D/3D simultaneously. Select Y-Slice for TYPE. '2.5*SIN(X-Y)' for EQ. 'X' for INDEP. Press NXT OK.

- ← WIN , simultaneously (in RPN mode). Use the following values:



```
PLOT WINDOW - Y-SLICE
X-Left:-5.          X-Right:5.
Y-Near:-5.          Y-Far:  5.
Z-Low: -2.5         Z-High: 2.5

Step Indep:20.      Depnd:20.

Enter minimum X view-volume val
```

- Press ERASE DRAW. Allow some time for the calculator to generate all the needed graphics. When ready, it will show a traveling sinusoidal wave in your screen.

## Animating a collection of graphics

The calculator provides the function ANIMATE to animate a number of graphics that have been placed in the stack. You can generate a graph in the graphics screen by using the commands in the PLOT and PICT menus. To place the generated graph in the stack, use PICT RCL. When you have *n* graphs in levels *n* through *1* of the stack, you can simply use the command *n* ANIMATE to produce an animation made of the graphs you placed in the stack.

Example 1 – Animating a ripple in a water surface

As an example, type in the following program that generates 11 graphics showing a circle centered in the middle of the graphics screen and whose radius increase by a constant value in each subsequent graph.

```
«                                    Begin program
RAD                                  Set angle units to radians
131 R→B 64 R→B PDIM                  Set PICT to 131×64 pixels
0 100 XRNG 0 100 YRNG                Set x- and y-ranges to 0-100
1 11 FOR j                           Start loop with j = 1 .. 11
  ERASE                              Erase current PICT
  (50., 50.) '5*(j-1)' →NUM          Centers of circles (50,50)
  0 '2*π' →NUM ARC                   Draw circle center r = 5(j-1)
  PICT RCL                           Place current PICT on stack
NEXT                                 End FOR-NEXT loop
11 ANIMATE                           Animate
»                                    End program
```

Store this program in a variable called PANIM (Plot ANIMation). To run the program press VAR (if needed) PANIM. It takes the calculator more than one minute to generate the graphs and get the animation going. Therefore, be really patient here. You will see the hourglass symbol up in the screen for what seems a long time before the animation, resembling the ripples produced by a pebble dropped on the surface of a body of quiescent water, appears in the screen. To stop the animation, press ON.

The 11 graphics generated by the program are still available in the stack. If you want to re-start the animation, simply use: 11 ANIMATE. (Function ANIMATE is available by using ← PRG NXT GROB NXT ANIM). The animation will be re-started. Press ON to stop the animation once more. Notice that the number 11 will still be listed in stack level 1. Press ◄ to drop it from the stack.

Suppose that you want to keep the figures that compose this animation in a variable. You can create a list of these figures, let's call it WLIST, by using:

/ / ← PRG TYPE →LIST ' ALPHA ALPHA W L I S T ALPHA STO►

Press VAR to recover your list of variables. The variable WLIST should now be listed in your soft-menu keys. To re-animate this list of variables you could use the following program:

```
«                          Start program
WLIST                      Place list WLIST in stack
OBJ→                       Decompose list, stack level 1 = 11
ANIMATE                    Start animation
»                          End program
```

Save this program in a variable called RANIM (Re-ANIMate). To run it, press RANIM.

The following program will animate the graphics in WLIST forward and backwards:

```
«                          Start program
WLIST   DUP                Place list WLIST in stack, make extra copy
REVLIST  +                 Reverse order, concatenate 2 lists
OBJ→                       Decompose list in elements, level 1 = 22
ANIMATE                    Start animation
»                          End program
```

Save this program in a variable called RANI2 (Re-ANImate version 2). To run it, press RANI2. The animation now simulates a ripple in the surface of otherwise quiescent water that gets reflected from the walls of a circular tank back towards the center. Press ON to stop the animation.

Example 2 - Animating the plotting of different power functions
Suppose that you want to animate the plotting of the functions f(x) = $x^n$, n = 0, 1, 2, 3, 4, in the same set of axes. You could use the following program:

```
«                                Begin program
RAD                              Set angle units to radians
131 R→B 64 R→B PDIM              Set PICT screen to 131×64 pixels
0 2 XRNG 0 20 YRNG               Set x- and y-ranges
0 4 FOR j                        Start loop with j = 0,1,…,4
  'X^j'  STEQ                    Store 'X^j' in variable EQ
  ERASE                          Erase current PICT
  DRAX LABEL DRAW                Draw axes, labels, function
  PICT RCL                       Place current PICT on stack
NEXT                             End FOR-NEXT loop
5 ANIMATE                        Animate
»
```

Store this program in a variable called PWAN (PoWer function ANimation). To run the program press VAR (if needed) PWAN. You will see the calculator drawing each individual power function before starting the animation in which the five functions will be plotted quickly one after the other. To stop the animation, press ON.

## More information on the ANIMATE function

The ANIMATE function as used in the two previous examples utilized as input the graphics to be animated and their number. You can use additional information to produce the animation, such as the time interval between graphics and the number of repetitions of the graphics. The general format of the ANIMATE function in such cases is the following:

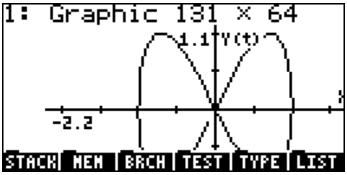
```
        n-graphs   { n {#X #Y} delay rep }  ANIMATE
```

`n` represents the number of graphics, `{#X #Y}` stand for the pixel coordinates of the lower right corner of the area to be plotted (see figure below), `delay` is the number of seconds allowed between consecutive graphics in the animation, and `rep` is the number of repetitions of the animation.

# Graphic objects (GROBs)

The word GROB stands for GRaphics OBjects and is used in the calculator's environment to represent a pixel-by-pixel description of an image that has been produced in the calculator's screen. Therefore, when an image is converted into a GROB, it becomes a sequence of binary digits (*b*inary dig*its* = *bits*), i.e., 0's and 1's. To illustrate GROBs and conversion of images to GROBS consider the following exercise.

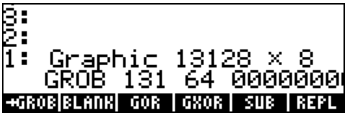
When we produce a graph in the calculator, the graph become the contents of a special variable called PICT. Thus, to see the last contents of PICT, you could use: PICT RCL (← PRG NXT PICT PICT ← RCL ).
The display shows in stack level 1 the `line Graphic 131×64` (if using the standard screen size) followed by a sketch of the top part of the graph. For example,



```
1: Graphic 131 × 64
```

If you press ▼ then the graph contained in level 1 is shown in the calculator's graphics display. Press CANCL to return to normal calculator display.

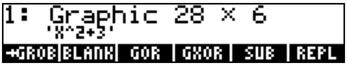
The graph in level 1 is still not in GROB format, although it is, by definition, a graphics object. To convert a graph in the stack into a GROB, use: 3 ENTER ← PRG NXT GROB →GROB. Now we have the following information in level 1:



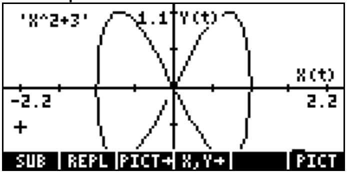
```
3:
2:
1: Graphic 13128 × 8
  GROB 131 64 0000000
```

The first part of the description is similar to what we had originally, namely, `Graphic 131×64`, but now it is expressed as `Graphic 13128 × 8`. However, the graphic display is now replaced by a sequence of zeroes and ones representing the pixels of the original graph. Thus, the original graph as now been converted to its equivalent representation in bits.

You can also convert equations into GROBs. For example, using the equation writer type in the equation 'X^2+3' into stack level 1, and then press / ENTER ← PRG NXT GROB →GROB . You will now have in level 1 the GROB described as:


```
1: Graphic 28 × 6
   'X^2+3'
```

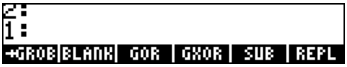
As a graphic object this equation can now be placed in the graphics display. To recover the graphics display press ◄. Then, move the cursor to an empty sector in the graph, and press EDIT NXT NXT REPL. The equation 'X^2-5' is placed in the graph, for example:

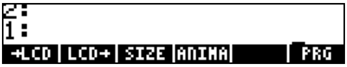


Thus, GROBs can be used to document graphics by placing equations, or text, in the graphics display.

## The GROB menu

The GROB menu, accessible through ← PRG NXT GROB →GROB, contains the following functions. Press NXT to move to the next menu:


```
2:
1:
```


```
2:
1:
```

### →GROB

Of these functions we have already used SUB, REPL, (from the graphics EDIT menu), ANIMATE [ANIMA], and →GROB. ([ *PRG* ] is simply a way to return to the programming menu.) While using →GROB in the two previous examples you may have noticed that I used a 3 while converting the graph into a GROB, while I used a 1 when I converted the equation into a GROB. This parameter of the function →GROB indicates the size of the object that is being converted into a GROB as 0 or 1 – for a small object, 2 – medium, and 3 – large. The other functions in the GROB menu are described following.

### BLANK

The function BLANK, with arguments #n and #m, creates a blank graphics object of width and height specified by the values #n and #m, respectively. This is similar to the function PDIM in the GRAPH menu.

### GOR

The function GOR (Graphics OR) takes as input *grob*$_2$ (a target GROB), a set of coordinates, and *grob*$_1$, and produces the superposition of *grob*$_1$ onto *grob*$_2$ (or PICT) starting at the specified coordinates. The coordinates can be specified as user-defined coordinates (x,y), or pixels {#n #m}. GOR uses the OR function to determine the status of each pixel (i.e., on or off) in the overlapping region between *grob*$_1$ and *grob*$_2$.

### GXOR

The function GXOR (Graphics XOR) performs the same operation as GOR, but using XOR to determine the final status of pixels in the overlapping area between graphic objects *grob*$_1$ and *grob*$_2$.

**Note**: In both GOR and GXOR, when *grob2* is replaced by PICT, these functions produce no output. To see the output you need to recall PICT to the stack by using either PICT RCL or PICTURE.

### →LCD

Takes a specified GROB and displays it in the calculator's display starting at the upper left corner.

### LCD→

Copies the contents of the stack and menu display into a 131 x 64 pixels GROB.

### SIZE

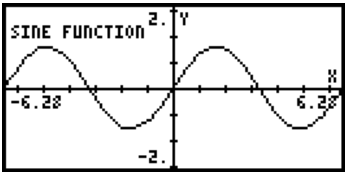
The function SIZE, when applied to a GROB, shows the GROB's size in the form of two numbers. The first number, shown in stack level 2, represents the width of the graphics object, and the second one, in stack level 1, shows its height.

### An example of a program using GROB

The following program produces the graph of the sine function including a frame – drawn with the function BOX – and a GROB to label the graph. Here is the listing of the program:

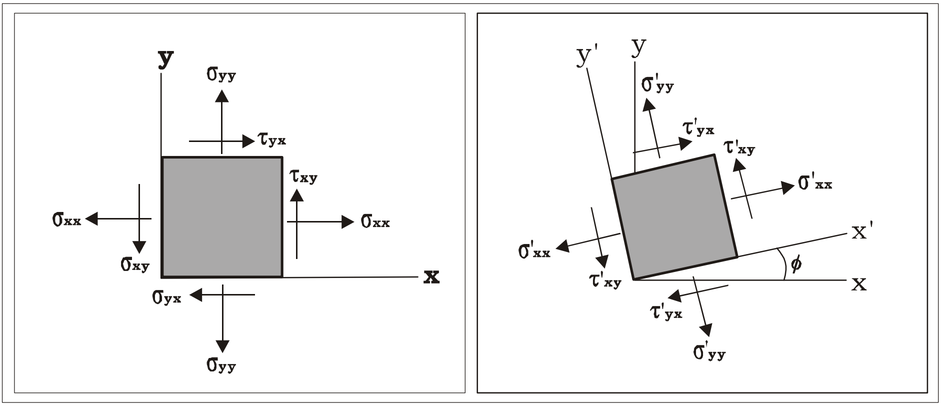
```
«                                    Begin program
RAD                                  Set angle units to radians
131 R→B 64 R→B PDIM                  Set PICT screen to 131×64 pixels
-6.28 6.28 XRNG –2. 2. YRNG          Set x- and y-ranges
FUNCTION                             Select FUNCTION type for graphs
'SIN(X)' STEQ                        Store the function sine into EQ
ERASE DRAX LABEL DRAW                Clear, draw axes, labels, graph
(-6.28,-2.) (6.28,2.) BOX            Draw a frame around the graph
PICT RCL                             Place contents of PICT on stack
"SINE FUNCTION"                      Place graph label string in stack
1 →GROB                              Convert string into a small GROB
(-6., 1.5) SWAP                      Coordinates to place label GROB
GOR                                  Combine PICT with the label GROB
PICT STO                             Save combined GROB into PICT
{ } PVIEW                            Bring PICT to the stack
»                                    End program
```

Save the program under the name GRPR (GROB PRogram). Press GRPR to run the program. The output will look like this:



## A program with plotting and drawing functions

In this section we develop a program to produce, draw and label Mohr's circle for a given condition of two-dimensional stress. The left-hand side figure below shows the given state of stress in two-dimensions, with $\sigma_{xx}$ and $\sigma_{yy}$ being normal stresses, and $\tau_{xy} = \tau_{yx}$ being shear stresses. The right-hand side figure shows the state of stresses when the element is rotated by an angle $\phi$. In this case, the normal stresses are $\sigma'_{xx}$ and $\sigma'_{yy}$, while the shear stresses are $\tau'_{xy}$ and $\tau'_{yx}$.

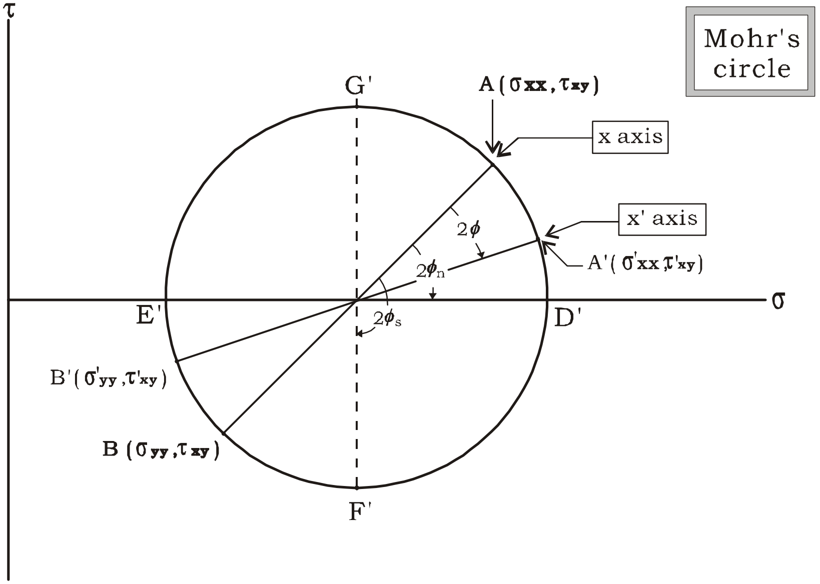


The relationship between the original state of stresses ($\sigma_{xx}$, $\sigma_{yy}$, $\tau_{xy}$, $\tau_{yx}$) and the state of stress when the axes are rotated counterclockwise by f ($\sigma'_{xx}$, $\sigma'_{yy}$, $\tau'_{xy}$, $\tau'_{yx}$), can be represented graphically by the construct shown in the figure below.

To construct Mohr's circle we use a Cartesian coordinate system with the x-axis corresponding to the normal stresses ($\sigma$), and the y-axis corresponding to the shear stresses ($\tau$). Locate the points A($\sigma_{xx}$,$\tau_{xy}$) and B ($\sigma_{yy}$, $\tau_{xy}$), and draw the segment AB. The point C where the segment AB crosses the $\sigma_n$ axis will be the center of the circle. Notice that the coordinates of point C are (½·($\sigma_{yy}$ + $\sigma_{xy}$), 0). When constructing the circle by hand, you can use a compass to trace the circle since you know the location of the center C and of two points, A and B.

Let the segment AC represent the x-axis in the original state of stress. If you want to determine the state of stress for a set of axes x'-y', rotated counterclockwise by an angle $\phi$ with respect to the original set of axes x-y, draw segment A'B', centered at C and rotated clockwise by and angle $2\phi$

with respect to segment AB. The coordinates of point A′ will give the values ($\sigma'_{xx}$,$\tau'_{xy}$), while those of B′ will give the values ($\sigma'_{yy}$,$\tau'_{xy}$).



The stress condition for which the shear stress, $\tau'_{xy}$, is zero, indicated by segment D′E′, produces the so-called *principal stresses*, $\sigma^{P}_{xx}$ (at point D′) and $\sigma^{P}_{yy}$ (at point E′). To obtain the principal stresses you need to rotate the coordinate system x′-y′ by an angle $\phi_n$, counterclockwise, with respect to the system x-y. In Mohr's circle, the angle between segments AC and D′C measures $2\phi_n$.

The stress condition for which the shear stress, $\tau'_{xy}$, is a maximum, is given by segment F′G′. Under such conditions both normal stresses, $\sigma'_{xx} = \sigma'_{yy}$ , are equal. The angle corresponding to this rotation is $\phi_s$. The angle between segment AC and segment F′C in the figure represents $2\phi_s$.

## Modular programming

To develop the program that will plot Mohr's circle given a state of stress, we will use modular programming. Basically, this approach consists in decomposing the program into a number of sub-programs that are created as separate variables in the calculator. These sub-programs are then linked by a main program, that we will call *MOHRCIRCL*. We will first create a sub-directory called MOHRC within the HOME directory, and move into that directory to type the programs.

The next step is to create the main program and sub-programs within the sub-directory.

The main program MOHRCIRCL uses the following sub-programs:

- INDAT: Requests input of σx, σy, τxy from user, produces a list σL = {σx, σy, τxy} as output.
- CC&r: Uses σL as input, produces σc = ½(σx+σy), r = radius of Mohr's circle, ϕn = angle for principal stresses, as output.
- DAXES: Uses σc and r as input, determines axes ranges and draws axes for the Mohr's circle construct
- PCIRC: Uses σc, r, and ϕn as input, draw's Mohr's circle by producing a PARAMETRIC plot
- DDIAM: Uses σL as input, draws the segment AB (see Mohr's circle figure above), joining the input data points in the Mohr's circle
- σLBL: Uses σL as input, places labels to identify points A and B with labels "σx" and "σy".
- σAXS: Places the labels "σ" and "τ" in the x- and y-axes, respectively.
- PTTL: Places the title "Mohr's circle" in the figure.

The programs are available in the diskette or CD ROM that comes with your calculator.

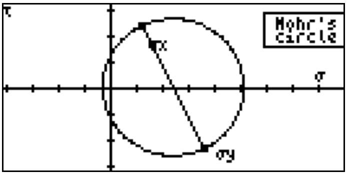
## Running the program

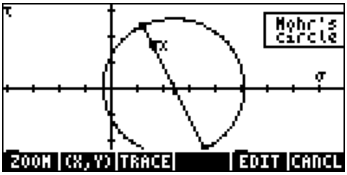
If you typed the programs in the order shown above, you will have in your sub-directory MOHRC the following variables: PTTL, σAXS, PLPNT, σLBL, PPTS, DDIAM. Pressing NXT you find also: PCIRC, DAXES, ATN2, CC&r,

INDAT, MOHRC. Before re-ordering the variables, run the program once by pressing the soft-key labeled MOHRC. Use the following:

| | |
|---|---|
| MOHRC | Launches the main program MOHRCIRCL |
| 2 5 ▼ | Enter $\sigma x = 25$ |
| 7 5 ▼ | Enter $\sigma y = 75$ |
| 5 0 ENTER | Enter $\tau xy = 50$, and finish data entry. |

At this point the program MOHRCIRCL starts calling the sub-programs to produce the figure. Be patient. The resulting Mohr's circle will look as in the picture to the left.





Because this view of PICT is invoked through the function PVIEW, we cannot get any other information out of the plot besides the figure itself. To obtain additional information out of the Mohr's circle, end the program by pressing ON. Then, press ◀ to recover the contents of PICT in the graphics environment. The Mohr's circle now looks like the picture to the right (see above).

Press the soft-menu keys TRACE and (X,Y). At the bottom of the screen you will find the value of $\phi$ corresponding to the point A($\sigma x$, $\tau xy$), i.e., `φ = 0, (2.50E1, 5.00E1).`

Press the right-arrow key (▶) to increment the value of $\phi$ and see the corresponding value of $(\sigma'_{xx}, \tau'_{xy})$. For example, for $\phi = 45°$, we have the values $(\sigma'_{xx}, \tau'_{xy})$ = (1.00E2, 2.50E1) = (100, 25). The value of $\sigma'_{yy}$ will be found at an angle 90° ahead, i.e., where $\phi = 45 + 90 = 135°$. Press the ▶ key until reaching that value of $\phi$, we find $(\sigma'_{yy}, \tau'_{xy})$ = (-1.00E-10,-2.5E1) = (0, 25).

To find the principal normal values press ◄ until the cursor returns to the intersection of the circle with the positive section of the σ-axis. The values found at that point are $\phi$ = 59°, and ($\sigma'_{xx}$, $\tau'_{xy}$) = (1.06E2,-1.40E0) = (106, -1.40). Now, we expected the value of $\tau'_{xy}$ = 0 at the location of the principal axes. What happens is that, because we have limited the resolution on the independent variable to be $\Delta\phi$ = 1°, we miss the actual point where the shear stresses become zero. If you press ◄ once more, you find values of are $\phi$ = 58°, and ($\sigma'_{xx}$, $\tau'_{xy}$) = (1.06E2,5.51E-1) = (106, 0.551). What this information tell us is that somewhere between $\phi$ = 58° and $\phi$ = 59°, the shear stress, $\tau'_{xy}$, becomes zero.

To find the actual value of $\phi$n, press ON. Then type the list corresponding to the values {σx σy τxy}, for this case, it will be `{ 25 75 50 } [ENTER]`

Then, press CC&r. The last result in the output, 58.2825255885°, is the actual value of $\phi$n.

## A program to calculate principal stresses

The procedure followed above to calculate $\phi$n, can be programmed as follows:

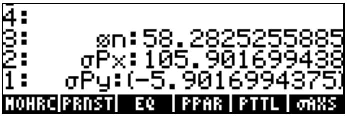
*Program PRNST:*

| | |
|---|---|
| « | Start program PRNST (PRiNcipal STresses) |
| INDAT | Enter data as in program MOHRCIRC |
| CC&r | Calculate σc, r, and fn, as in MOHRCIRC |
| "$\phi$n" →TAG | Tag angle for principal stresses |
| 3 ROLLD | Move tagged angle to level 3 |
| R→C DUP | Convert σc and r to (σc, r), duplicate |
| C→R + "σPx" →TAG | Calculate principal stress σPx, tag it |
| SWAP C→R - "σPy" →TAG | Swap,calculate stress σPy, tag it. |
| » | End program PRNST |

To run the program use:

| | |
|---|---|
| VAR PRNST | Start program PRNST |
| 2 5 ▼ | Enter σx = 25 |
| 7 5 ▼ | Enter σy = 75 |
| 5 0 ENTER | Enter τxy = 50, and finish data entry. |

The result is:



```
4:
3:     ⌀n:58.2825255885
2:    σPx:105.901699438
1:  σPy:(-5.9016994375)
MOHRC|PRNST| EQ |PPAR|PTTL|σAXS
```

## Ordering the variables in the sub-directory

Running the program MOHRCIRCL for the first time produced a couple of new variables, PPAR and EQ. These are the Plot PARameter and EQuation variables necessary to plot the circle. It is suggest that we re-order the variables in the sub-directory, so that the programs MOHRC and PRNST are the two first variables in the soft-menu key labels. This can be accomplished by creating the list { MOHRCIRCL PRNST } using: VAR ← {} MOHRC PRNST ENTER
And then, ordering the list by using: ← PRG MEM DIR ORDER.

After this call to the function ORDER is performed, press VAR. You will now see that we have the programs MOHRCIRCL and PRNST being the first two variables in the menu, as we expected.

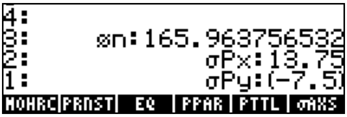
## A second example of Mohr's circle calculations

Determine the principal stresses for the stress state defined by $\sigma_{xx}$ = 12.5 kPa, $\sigma_{yy}$ = -6.25 kPa, and $\tau_{xy}$ = - 5.0 kPa. Draw Mohr's circle, and determine from the figure the values of $\sigma'_{xx}$, $\sigma'_{yy}$, and $\tau'_{xy}$ if the angle $\phi$ = 35°.

To determine the principal stresses use the program PRNST, as follows:

| | |
|---|---|
| VAR PRNST | Start program PRNST |
| 1 2 . 5 ▼ | Enter σx = 12.5 |
| 6 . 2 5 +/- ▼ | Enter σy = -6.25 |
| 5 +/- ENTER | Enter τxy = -5, and finish data entry. |

The result is:

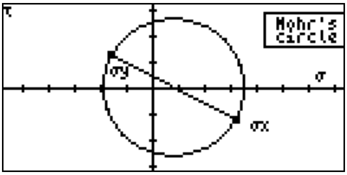


```
4:
3:     ⌀n:165.963756532
2:              σPx:13.75
1:             σPy:(-7.5)
MOHRC|PRNST| EQ |PPAR|PTTL|σAXS
```

To draw Mohr's circle, use the program MOHRC, as follows:

| | |
|---|---|
| VAR MOHRC | Start program PRNST |
| 1 2 • 5 ▼ | Enter σx = 12.5 |
| 6 • 2 5 +/- ▼ | Enter σy = -6.25 |
| 5 +/- ENTER | Enter τxy = -5, and finish data entry. |

The result is:



To find the values of the stresses corresponding to a rotation of 35° in the angle of the stressed particle, we use:

| | |
|---|---|
| ON ◄ | Clear screen, show PICT in graphics screen |
| TRACE (X,Y). | To move cursor over the circle showing $\phi$ and (x,y) |

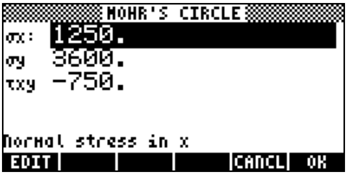
Next, press ► until you read $\phi$ = 35. The corresponding coordinates are (1.63E0, -1.05E1), i.e., at $\phi$ = 35°, $\sigma'_{xx}$ = 1.63 kPa, and $\sigma'_{yy}$ = -10.5kPa.

## An input form for the Mohr's circle program

For a fancier way to input data, we can replace sub-program INDAT, with the following program that activates an input form:

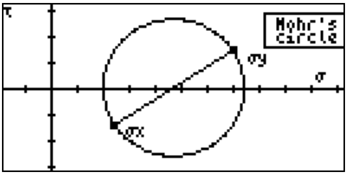
```
« “MOHR’S CIRCLE” { { “σx:” “Normal stress in x” 0 }
{ “σy:” “Normal stress in y” 0 } { “τxy:” “Shear stress”
0} }   { } { 1 1 1 } { 1 1 1 }  INFORM DROP »
```

With this program substitution, running MOHRC will produce an input form as shown next:

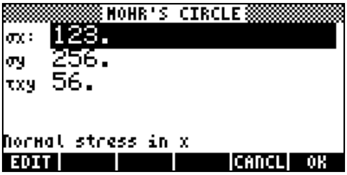

MOHR'S CIRCLE
σx: 1250.
σy 3600.
τxy -750.

Normal stress in x

Press OK to continue program execution. The result is the following figure:

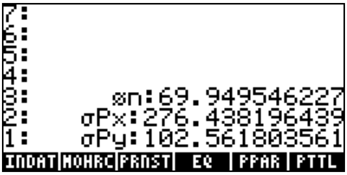


Since program INDAT is used also for program PRNST (PRiNcipal STresses), running that particular program will now use an input form, for example,



```
MOHR'S CIRCLE
σx: 123.
σy  256.
τxy 56.

Normal stress in x
```

The result, after pressing OK, is the following:



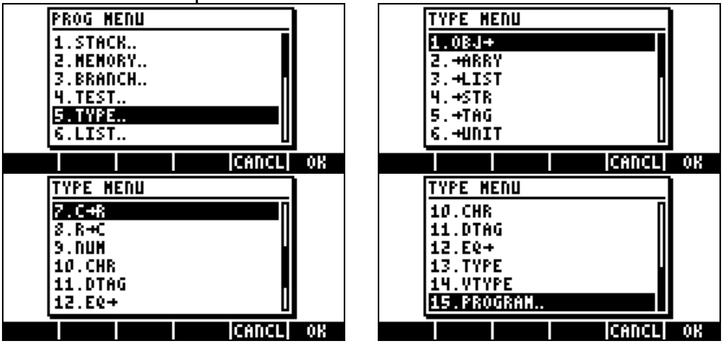
```
7:
6:
5:
4:
3:      θn:69.949546227
2:    σPx:276.438196439
1:    σPy:102.561803561
```

# Chapter 23
# Character strings

Character strings are calculator objects enclosed between double quotes. They are treated as text by the calculator. For example, the string "SINE FUNCTION", can be transformed into a GROB (Graphics Object), to label a graph, or can be used as output in a program. Sets of characters typed by the user as input to a program are treated as strings. Also, many objects in program output are also strings.

## String-related functions in the TYPE sub-menu

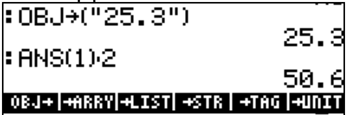
The TYPE sub-menu is accessible through the PRG (programming) menu, i.e., ← PRG . The functions provided in the TYPE sub-menu are also shown below.



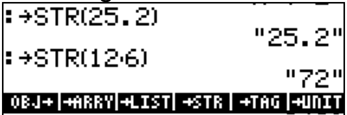
Among the functions in the TYPE menu that are useful for manipulating strings we have:

OBJ→: Converts string to the object it represents
→STR: Converts an object to its string representation
→TAG: Tags a quantity
DTAG: Removes the tag from a tagged quantity (de-tags)
CHR: Creates a one-character string corresponding to the number used as argument
NUM: Returns the code for first character in a string

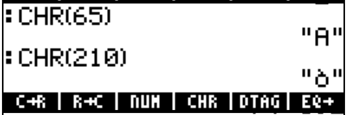
Examples of application of these functions to strings are shown next:



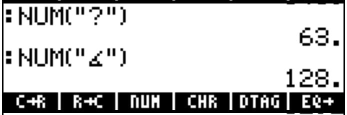
```
:OBJ→("25.3")
                  25.3
:ANS(1)·2
                  50.6
```



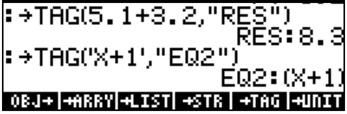
```
:→STR(25.2)
                "25.2"
:→STR(12·6)
                  "72"
```



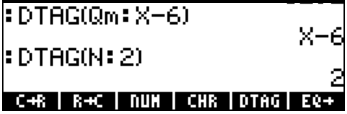
```
:CHR(65)
                   "A"
:CHR(210)
                   "ò"
```



```
:NUM("?")
                   63.
:NUM("∡")
                  128.
```



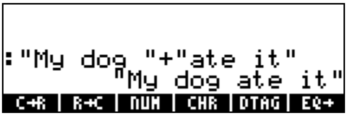
```
:→TAG(5.1+3.2,"RES")
               RES:8.3
:→TAG('X+1',"EQ2")
              EQ2:(X+1)
```



```
:DTAG(Qm:X-6)
                   X-6
:DTAG(N:2)
                     2
```

## String concatenation

Strings can be concatenated (joined together) by using the plus sign +, for example:

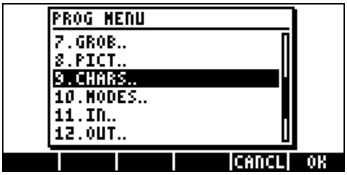


```
:"My dog "+"ate it"
        "My dog ate it"
```

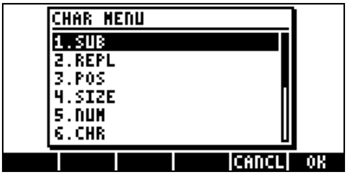
Concatenating strings is a practical way to create output in programs. For example, concatenating "YOU ARE " AGE + " YEAR OLD" creates the string "YOU ARE 25 YEAR OLD", where 25 is stored in the variable called AGE.

## The CHARS menu

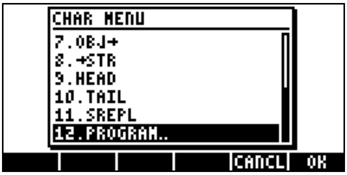
The CHARS sub-menu is accessible through the PRG (programming) menu, i.e., ← PRG .



The functions provided by the CHARS sub-menu are the following:



```
CHAR MENU
1.SUB
2.REPL
3.POS
4.SIZE
5.NUM
6.CHR
```

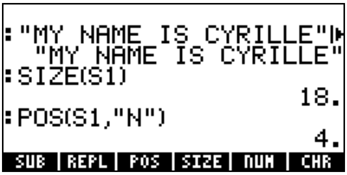


```
CHAR MENU
7.OBJ→
8.→STR
9.HEAD
10.TAIL
11.SREPL
12.PROGRAM..
```

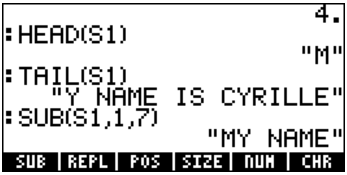
The operation of NUM, CHR, OBJ→, and →STR was presented earlier in this Chapter. We have also seen the functions SUB and REPL in relation to graphics earlier in this chapter. Functions SUB, REPL, POS, SIZE, HEAD, and TAIL have similar effects as in lists, namely:

SIZE: number of a sub-string in a string (including spaces)
POS: position of first occurrence of a character in a string
HEAD: extracts first character in a string
TAIL: removes first character in a string
SUB: extract sub-string given starting and ending position
REPL: replace characters in a string with a sub-string starting at given position
SREPL: replaces a sub-string by another sub-string in a string

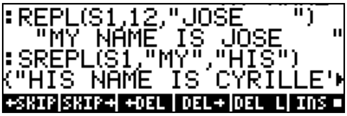
To see those effects on action try the following exercises: Store the string "MY NAME IS CYRILLE" into variable S1. We'll use this string to show examples of the functions in the CHARS menu:



```
:"MY NAME IS CYRILLE"▸
  "MY NAME IS CYRILLE"
:SIZE(S1)
                  18.
:POS(S1,"N")
                   4.
```



```
                   4.
:HEAD(S1)
                  "M"
:TAIL(S1)
   "Y NAME IS CYRILLE"
:SUB(S1,1,7)
            "MY NAME"
```



```
:REPL(S1,12,"JOSE    ")
  "MY NAME IS JOSE    "
:SREPL(S1,"MY","HIS")
{"HIS NAME IS CYRILLE"▸
```

## The characters list

The entire collection of characters available in the calculator is accessible through the keystroke sequence ↱ CHARS When you highlight any character, say they line feed character ↵ , you will see at the left side of the bottom of the screen the keystroke sequence to get such character (↱. for this case) and the numerical code corresponding to the character (10 in this case).

Characters that are not defined appear as a dark square in the characters list (▪) and show (None) at the bottom of the display, even though a numerical code exists for all of them. Numerical characters show the corresponding number at the bottom of the display.

Letters show the code α (i.e., ALPHA) followed by the corresponding letter, for example, when you highlight M, you will see αM displayed at the lower left side of the screen, indicating the use of ALPHA M . On the other hand, m shows the keystroke combination α↰M, or ALPHA ↰ M .

Greek characters, such as σ, will show the code α↱S, or ALPHA ↱ S . Some characters, like ρ, do not have a keystroke sequence associated with them. Therefore, the only way to obtain such characters is through the character list by highlighting the desired character and pressing ECHO1 or ECHO.

Use ECHO1 to copy one character to the stack and return immediately to normal calculator display. Use ECHO to copy a series of characters to the stack. To return to normal calculator display use ON .

See Appendix D for more details on the use of special characters. Also, Appendix G shows shortcuts for producing special characters.

# Chapter 24
# Calculator objects and flags

Numbers, lists, vectors, matrices, algebraics, etc., are calculator objects. They are classified according to its nature into 30 different types, which are described below. Flags are variables that can be used to control the calculator properties. Flags were introduced in Chapter 2.

## Description of calculator objects

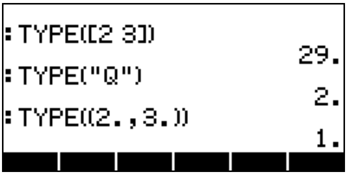
The calculator recognizes the following object types:

| Number | Type | Example |
|---|---|---|
| 0 | Real Number | -1.23E-5 |
| 1 | Complex Number | (-1.2,2.3) |
| 2 | String | "Hello, world " |
| 3 | Real Array | [[1 2][3 4]] |
| 4 | Complex Array | [[(1 2) (3 4)]<br>[(5 6) (7 8)]] |
| 5 | List | {3 1 'PI'} |
| 6 | Global Name | X |
| 7 | Local Name | y |
| 8 | Program | << → a 'a^2' >> |
| 9 | Algebraic object | 'a^2+b^2' |
| 10 | Binary Integer | # A2F1E h |
| 11 | Graphic Object | Graphic 131×64 |
| 12 | Tagged Object | R: 43.5 |
| 13 | Unit Object | 3_m^2/s |
| 14 | XLIB Name | XLIB 342 8 |
| 15 | Directory | DIR Σ END |
| 16 | Library | Library 1230"... |
| 17 | Backup Object | Backup MYDIR |
| 18 | Built-in Function | COS |
| 19 | Built-in Command | CLEAR |

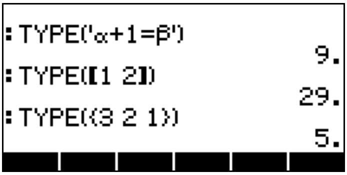
| Number | Type | Example |
|---|---|---|
| 21 | Extended Real Number | Long Real |
| 22 | Extended Complex Number | Long Complex |
| 23 | Linked Array | Linked Array |
| 24 | Character Object | Character |
| 25 | Code Object | Code |
| 26 | Library Data | Library Data |
| 27 | External Object | External |
| 28 | Integer | 3423142 |
| 29 | External Object | External |
| 30 | External Object | External |

## Function TYPE

This function, available in the PRG/TYPE () sub-menu, or through the command catalog, is used to determine the type of an object. The function argument is the object of interest. The function returns the object type as indicated by the numbers specified above.



```
:TYPE([2 3])
                 29.
:TYPE("Q")
                  2.
:TYPE((2.,3.))
                  1.
```



```
:TYPE('α+1=β')
                  9.
:TYPE([[1 2]])
                 29.
:TYPE({3 2 1})
                  5.
```

## Function VTYPE

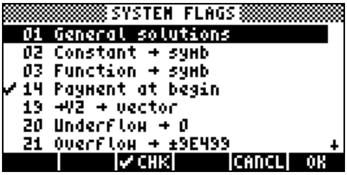
This function operates similar to function TYPE, but it applies to a variable name, returning the type of object stored in the variable.

# Calculator flags

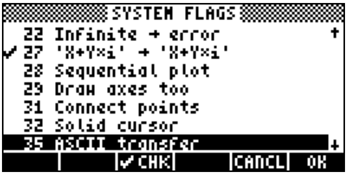
A flag is a variable that can either be set or unset. The status of a flag affects the behavior of the calculator, if the flag is a system flag, or of a program, if it is a user flag. They are described in more detail next.

## System flags

System flags can be accessed by using MODE FLAGS. Press the down arrow key to see a listing of all the system flags with their number and brief description. The first two screens with system flags are shown below:



```
SYSTEM FLAGS
  01 General solutions
  02 Constant → symb
  03 Function → symb
✓ 14 Payment at begin
  19 →V2 → vector
  20 Underflow → 0
  21 Overflow → ±9E499
```



```
SYSTEM FLAGS
  22 Infinite → error
✓ 27 'X+Y×i' → 'X+Y×i'
  28 Sequential plot
  29 Draw axes too
  31 Connect points
  32 Solid cursor
  35 ASCII transfer
```

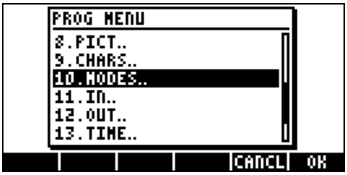
You will recognize many of these flags because they are set or unset in the MODES menu (e.g., flag 95 for Algebraic mode, 103 for Complex mode, etc.). Throughout this User' s Manual we have emphasized the differences between CHOOSE boxes and SOFT menus, which are selected by setting or un-setting system flag 117. Another example of system flag setting is that of system flags 60 and 61 that relate to the constant library (CONLIB, see Chapter 3). These flags operate in the following manner:

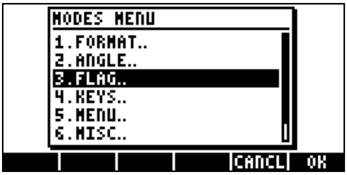
- user flag 60: clear (default):SI units, set: ENGL units
- user flag 61: clear (default):use units, set: value only

## Functions for setting and changing flags

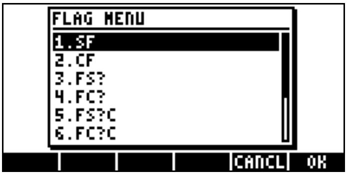
These functions can be used to set, un-set, or check on the status of user flags or system flags. When used with these functions system flags are referred to with negative integer numbers. Thus, system flag 117 will be referred to as flag -117. On the other hand, user flags will be referred to as positive integer numbers when applying these functions. It is important to understand that user flags have applications only in programming to help control the program flow.

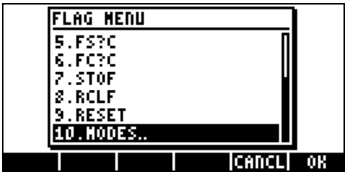
Functions for manipulating calculator flags are available in the PRG/MODES/FLAG menu. The PRG menu is activated with ← PRG . The following screens (with system flag 117 set to CHOOSE boxes) show the sequence of screens to get to the FLAG menu:





The functions contained within the FLAG menu are the following:





The operation of these functions is as follows:

| | |
|---|---|
| SF | Set a flag |
| CF | Clear a flag |
| FS? | Returns 1 if flag is set, 0 if not set |
| FC? | Returns 1 if flag is clear (not set), 0 if flag is set |
| FS?C | Tests flag as FS does, then clears it |
| FC?C | Tests flag as FC does, then clears it |
| STOF | Stores new system flag settings |
| RCLF | Recalls existing flag settings |
| RESET | Resets current field values (could be used to reset a flag) |

## User flags

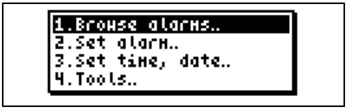
For programming purposes, flags 1 through 256 are available to the user. They have no meaning to the calculator operation.

# Chapter 25
# Date and Time Functions

In this Chapter we demonstrate some of the functions and calculations using times and dates.

## The TIME menu

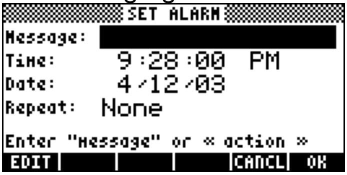
The TIME menu, available through the keystroke sequence ↱ TIME (the 9 key) provides the following functions, which are described next:



1.Browse alarms..
2.Set alarm..
3.Set time, date..
4.Tools..

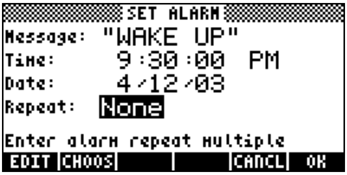
### Setting an alarm

Option *2. Set alarm..* provides an input form to let the user set an alarm. The input form looks like in the following figure:

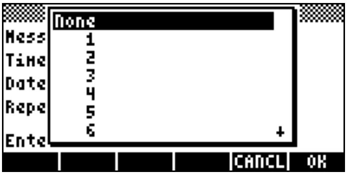


SET ALARM
Message:
Time: 9:28:00 PM
Date: 4/12/03
Repeat: None
Enter "message" or « action »

The Message: input field allows you to enter a character string identifying the alarm. The Time: field lets you enter the time for activating the alarm. The Date: field is used to set the date for the alarm (or for the first time of activation, if repetition is required). For example, you could set the following alarm. The left-hand side figure shows the alarm with no repetition. The right-hand figure shows the options for repetition after pressing CHOOS. After pressing OK the alarm will be set.

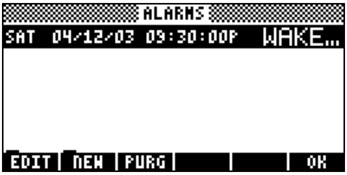


SET ALARM
Message: "WAKE UP"
Time: 9:30:00 PM
Date: 4/12/03
Repeat: None
Enter alarm repeat multiple



## Browsing alarms

Option *1. Browse alarms…* in the TIME menu lets you review your current alarms. For example, after entering the alarm used in the example above, this option will show the following screen:



SAT 04/12/03 09:30:00P WAKE…

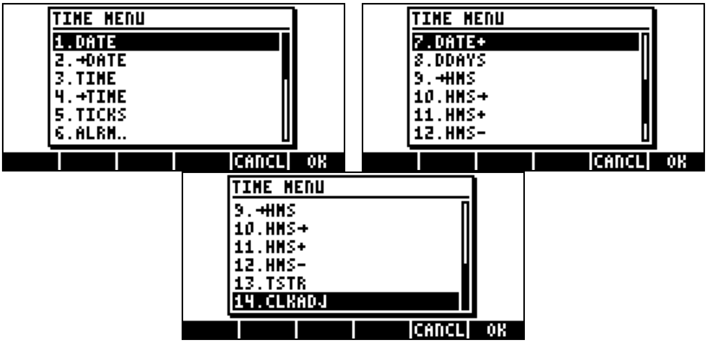
This screen provides four soft menu key labels:
EDIT: For editing the selected alarm, providing an alarm set input form
NEW: For programming a new alarm PURG: For deleting an alarm
OK : Returns to normal display

## Setting time and date

Option *3. Set time, date…* provides the following input form that let's the user set the current time and date. Details were provided in Chapter 1.

## TIME Tools

Option *4. Tools…* provides a number of functions useful for clock operation, and calculations with times and dates. The following figure shows the functions available under TIME Tools:

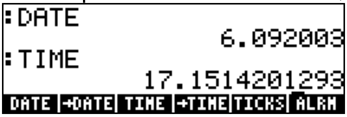


The application of these functions is demonstrated below.

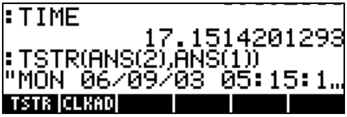
DATE: Places current date in the stack
→DATE: Set system date to specified value
TIME: Places current time in 24-hr HH.MMSS format
→TIME: Set system time to specified value in 24-hr HH.MM.SS format
TICKS: Provides system time as binary integer in units of clock ticks with 1 tick = 1/8192 sec
ALRM..:Sub-menu with alarm manipulation functions (described later)
DATE+: Adds or subtract a number of days to a date
DDAYS(x,y): Returns number of days between dates x and y
→HMS: Converts time from decimal to HH.MMSS
HMS→: Converts time from HH.MMSS to decimal
HMS+: Add two times in HH.MMSS format
HMS-: Subtract two times in HH.MMSS format
TSTR(time, date): converts time, date to string format
CLKADJ(x): Adds x ticks to system time (1 tick = 1/8192 sec )

Functions →DATE, →TIME, CLKADJ are used to adjust date and time. There are no examples provided here for these functions.

Here are examples of functions DATE, TIME, and TSTR:



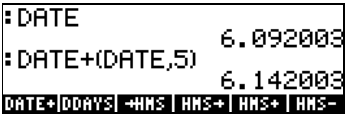
```
:DATE
                  6.092003
:TIME
              17.1514201293
```



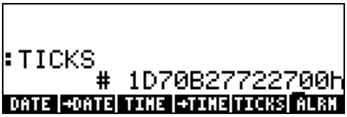
```
:TIME
              17.1514201293
:TSTR(ANS(2),ANS(1))
"MON 06/09/03 05:15:1…
```

## Calculations with dates

For calculations with dates, use functions DATE+, DDAYS. Here is an example of application of these functions, together with an example of function TICKS:



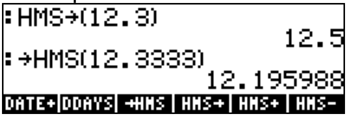
```
:DATE
                  6.092003
:DATE+(DATE,5)
                  6.142003
```



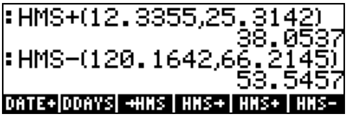
```
:TICKS
         # 1D70B27722700h
```

# Calculating with times

The functions →HMS, HMS→, HMS+, and HMS- are used to manipulate values in the HH.MMSS format. This is the same format used to calculate with angle measures in degrees, minutes, and seconds. Thus, these operations are useful not only for time calculations, but also for angular calculations. Examples are provided next:



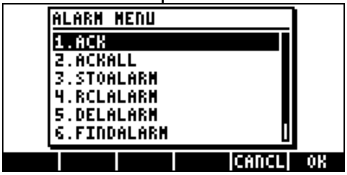
```
:HMS→(12.3)
                12.5
:→HMS(12.3333)
           12.195988
```



```
:HMS+(12.3355,25.3142)
             38.0537
:HMS-(120.1642,66.2145)
             53.5457
```

# Alarm functions

The sub-menu TIME/Tools…/ALRM… provides the following functions:



The operation of these functions is provided next:

ACK: Acknowledges past due alarm
ACKALL: Acknowledges all past due alarms
STOALARM(x): Stores alarm (x) into system alarm list
RCLALARM(x): Recalls specified alarm (x) from system alarm list
DELALARM(x): Deletes alarm x from system alarm list
FINDALARM(x): Returns first alarm due after specified time

The argument x in function STOALARM is a list containing a date reference (mm.ddyyy), time of day in 24 hr format (hh.mm), a string containing the text of the alarm, and the number of repetitions of the alarm. For example, `STOALARM({6.092003,18.25,"Test",0})`. The argument x in all the other alarm functions is a positive integer number indicating the number of the alarm to be recalled, deleted, or found.

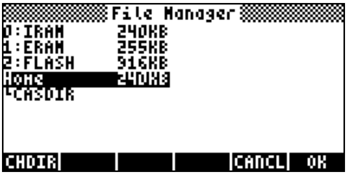
Since the handling of alarms can be easily done with the TIME menu (see above), the alarm-related functions in this section are more likely to be used for programming purposes.

# Chapter 26
# Managing memory

In Chapter 2 of the User's Guide we introduced the basic concepts and operations for creating and managing variables and directories. In this Chapter we discuss the management of the calculator's memory in terms of partition of memory and techniques for backing up data.

## Memory Structure

The calculator contains a total of 2.5 MB of memory, out of which 1 MB is used to store the operating system (system memory), and 1.5 MB is used for calculator operation and data storage (user's memory). Users do not have access to the system memory component. To see the way in which the user's memory is partitioned, use the FILES function (↰ FILES ). A possible result is shown below:



```
File Manager
0:IRAM      240KB
1:ERAM      255KB
2:FLASH     916KB
HOME        240KB
└CASDIR
```

This screen indicates the existence of three memory ports, besides the memory corresponding to the HOME directory (See Chapter 2 in the User's Guide). The memory ports available are:

- Port 0, labeled IRAM
- Port 1, labeled ERAM
- Port 2, labeled FLASH

Port 0 and the HOME directory share the same area of memory, so that the more data stored in the HOME directory, for example, the less memory is available for Port 0 storage. The total size of memory for the Port 0/HOME directory memory area is 241 KB.

Port 1 (ERAM) can contain up to 255 KB of data. Port 1, together with Port 0 and the HOME directory constitute the calculator's RAM (Random Access

Memory) segment of calculator's memory. The RAM memory segment requires continuous electric power supply from the calculator batteries to operate. To avoid loss of the RAM memory contents, a CR2032 backup battery is included. See additional details at the end of this chapter.

Port 2 belongs to the calculator's Flash ROM (Read-Only Memory) segment, which does not require a power supply. Therefore, removing the batteries of the calculators will not affect the calculator's Flash ROM segment. Port 2 can store up to 1085 KB of data.

### The HOME directory

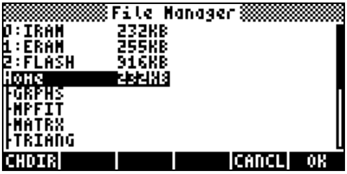
When using the calculator you may be creating variables to store intermediate and final results. Some calculator operations such as graphics or statistical operations create their own variables for storing data. These variables will be contained within the HOME directory or one of its directories. Details on the manipulation of variables and directories are presented in Chapter 2 of the User's Guide.

### Port memory

Unlike the HOME directory, port memory cannot be sub-divided into directories, and it can only contain backup objects or library objects. These object types are described below.

## Checking objects in memory

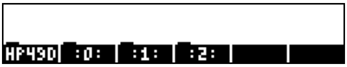
To see the objects stored in memory you can use the FILES function ( ↰ FILES ). The screen shows the HOME directory with at least four directories, namely, GRPHS, MPFIT, MATRX, and TRIANG.



```
File Manager
0:IRAM      232KB
1:ERAM      255KB
2:FLASH     916KB
HOME        232KB
├GRPHS
├MPFIT
├MATRX
├TRIANG
CHDIR            CANCL  OK
```

Additional directories can be viewed by moving the cursor downwards in the directory tree. Or you can move the cursor upwards to select a memory port. When a given directory, sub-directory or port is selected, press OK to see the contents of the selected object.

Another way to access port memory is by using the LIB menu (↱ LIB , associated with the 2 key). This action produces the following screen:



If you have any library active in your calculator it will be shown in this screen. One such library is the HP49D (demo) library shown in the screen above. Pressing the corresponding soft-menu key (F1) will activate this library. Pressing the port soft menu keys will open that memory port. Additional information on libraries is presented below.

## Backup objects

Backup objects are used to copy data from your home directory into a memory port. The purpose of backing up objects in memory port is to preserve the contents of the objects for future usage. Backup objects have the following characteristics:

- Backup objects can only exist in port memory (i.e., you cannot back up an object in the HOME directory, although you can make as many copies of it as you want)
- You cannot modify the contents of a backup object (you can, however, copy it back to a directory in the HOME directory, modify it there, and back it up again modified)
- You can store either a single object or an entire directory as a single backup object. You cannot, however, create a backup object out of a number of selected objects in a directory.

When you create a backup object in port memory, the calculator obtains a *cyclic redundancy check* (CRC) or *checksum* value based on the binary data contained in the object. This value is stored with the backup object, and is used by the calculator to monitor the integrity of the backup object. When you restore a backup object into the HOME directory, the calculator obtains again the CRC value and compares it to the original value. If a discrepancy is noticed, the calculator warns the user that the restored data may be corrupted.

# Backing up objects in port memory

The operation of backing up an object from user memory into one of the memory ports is similar to the operation of copying a variable from one sub-directory to another (see details in Chapter 2 of the User's Guide). You can, for example, use the File Manager (↰ FILES ) to copy and delete backup objects as you would do with normal calculator objects. In addition, there are specific commands for manipulating back up objects, as described next.

## Backing up and restoring HOME

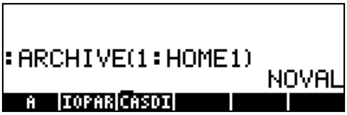
You can back up the contents of the current HOME directory in a single back up object. This object will contain all variables, key assignments, and alarms currently defined in the HOME directory. You can also restore the contents of your HOME directory from a back up object previously stored in port memory. The instructions for these operations follow.

### Backing up the HOME directory

To back up the current HOME directory using algebraic mode, enter the command:

ARCHIVE(:Port_Number: Backup_Name)

Here, Port_Number is 0, 1, 2 (or 3, if an SD memory card is available -- see below), and Backup_Name is the name of the backup object that will store the contents of HOME. The : : container is entered by using the keystroke sequence ↰ :: . For example, to back up HOME into HOME1 in Port 1, use:



```
:ARCHIVE(1:HOME1)
                NOVAL
```

To back up the HOME directory in RPN mode, use the command:

: Port_Number : Backup_Name ENTER ARCHIVE

### Restoring the HOME directory

To restore the home directory in algebraic mode use the command:

RESTORE(: Port_Number : Backup_Name)

For example, to restore the HOME directory out of backup object HOME1, use: `RESTORE(:1:HOME1)`

In RPN mode use:

: Port_Number : Backup_Name ENTER RESTORE

> **Note:** When you restore a HOME directory backup two things happen:
> - The backup directory overwrites the current HOME directory. Thus, any data not backed up in the current HOME directory will be lost.
> - The calculator restarts. The contents of history or stack are lost.

## Storing, deleting, and restoring backup objects

To create a backup object use one of the following approaches:

- Use the File Manager (← FILES ) to copy the object to port. Using this approach, the backup object will have the same name as the original object.
- Use the STO command to copy the object to a port. For example, in algebraic mode, to back up variable A into a backup object named AA in port 1, use the keystroke sequence:

  A STO▶ ← :: / ▶ ALPHA A ALPHA A ENTER
- Use the ARCHIVE command to create a backup of the HOME directory (see above).

To delete a backup object from a port:

- Use the File Manager (← FILES ) to delete the object as you would a variable in the HOME directory (see Chapter 2 in the User's Guide).
- Use the PURGE command as follows:

  In algebraic mode, use: PURGE(: Port_Number : Backup_Name)

  In RPN mode, use: : Port_Number : Backup_Name PURGE

To restore a backup object:

- Use the File Manager (← FILES ) to copy the backup object from Port memory to the HOME directory.

- When a backup object is restored, the calculator performs an integrity check on the restored object by calculating its CRC value. Any discrepancy between the calculated and the stored CRC values result in an error message indicating a corrupted data.

## Using data in backup objects

Although you cannot directly modify the contents of backup objects, you can use those contents in calculator operations. For example, you can run programs stored as backup objects or use data from backup objects to run programs. To run backup-object programs or use data from backup objects you can use the File Manager (↰ FILES ) to copy backup object contents to the screen. Alternatively, you can use function EVAL to run a program stored in a backup object, or function RCL to recover data from a backup object as follows:

- In algebraic mode:
  - To evaluate a back up object, enter:
    EVAL(argument(s), : Port_Number : Backup_Name )
  - To recall a backup object to the command line, enter:
    RCL(: Port_Number : Backup_Name)

- In RPN mode:
  - To evaluate a back up object, enter:
    Argument(s) ENTER : Port_Number : Backup_Name EVAL
  - To recall a backup object to the command line, enter:
    : Port_Number : Backup_Name ENTER RCL

## Using SD cards

The calculator provides a memory card port where you can insert an SD flash card for backing up calculator objects, or for downloading objects from other sources. Insert the card with the label side down. The SD card in the calculator will appear as port number 3.

Accessing an object from the SD card is performed similarly as if the object were located in ports 0, 1, or 2. However, Port 3 will not appear in the menu when using the LIB function (↱ LIB ). The SD files can only be managed using the Filer, or File Manager (↰ FILES ). When starting the Filer, the Tree view will show:

```
0: IRAM
1: ERAM
2: FLASH
3: SD
HOME
 |-- sub-directories
```

When you enter in the SD tree, all objects will appear as backup objects. Therefore, it is not possible to tell what type a given objects by just looking at its name in the Filer. Long names are not supported in the Filer. Thus, all names must be in the form 8.3 characters, similar as used in DOS, i.e., names will have a maximum of 8 characters with 3 characters in the suffix.

As an alternative to using the File Manager operations, you can use functions STO and RCL to store and recall objects from the SD card, as shown below. You can also use the PURGE command to erase backup objects in the SD card. Long names can be used with these commands (namely, STO, RCL, and PURGE).

## Storing objects in the SD card

You can only store an object at the root of the SD, i.e., no sub-directory tree can be build into Port 3 (This feature may be enhanced in a future flash ROM upgrade). To store an object, use function STO as follows:

- In algebraic mode:
  Enter object, press STO▶, type the name of the stored object using port 3 (e.g., :3:VAR1), press ENTER.
- In RPN mode:
  Enter object, type the name of the stored object using port 3 (e.g., :3:VAR1), press STO▶.

## Recalling an object from the SD card

To recall an object from the SD card onto the screen, use function RCL, as follows:

- In algebraic mode:
  Press ↰ RCL, type the name of the stored object using port 3 (e.g., :3:VAR1), press ENTER.

- In RPN mode:
  Type the name of the stored object using port 3 (e.g., :3:VAR1), press ← RCL .

With the RCL command, it is possible to recall variables by specifying a path in the command, e.g., in RPN mode: :3: {path} ENTER RCL. The path, like in a DOS drive, is a series of directory names that locate the position of the variable within a directory tree. However, some variables stored within a backup object cannot be recalled by specifying a path. In this case, the full backup object (e.g., a directory) will have to be recalled, and the individual variables then accessed in the screen.

### Purging an object from the SD card

To purge an object from the SD card onto the screen, use function PURGE, as follows:

- In algebraic mode:
  Press TOOL PURGE, type the name of the stored object using port 3 (e.g., :3:VAR1), press ENTER .
- In RPN mode:
  Type the name of the stored object using port 3 (e.g., :3:VAR1), press TOOL PURGE.

## Using libraries

Libraries are user-created binary-language programs that can be loaded into the calculator and made available for use from within any sub-directory of the HOME directory. Libraries can be downloaded into the calculator as a regular variable, and, then, installed and attached to the HOME directory.

### Installing and attaching a library

To install a library, list the library contents in the stack (use → variable soft-menu key, or function RCL) and store it into port 0 or 1. For example, to install a library variable into a port use:

- In algebraic mode: STO(Library_variable, port_number)
- In RPN mode: Library_variable ENTER port_number STO▸

After installing the library contents in port memory you need to attach the library to the HOME directory. This can be accomplished by rebooting the calculator (turning the calculator off and back on), or by pressing, simultaneously, ON F3 . At this point the library should be available for use. To see the library activation menu use the LIB menu (↱ LIB ). The library name will be listed in this menu.

## Library numbers

If you use the LIB menu (↱ LIB ) and press the soft menu key corresponding to port 0 or 1, you will see library numbers listed in the soft menu key labels. Each library has a four-digit number associated with it. These numbers are assigned by the library creator, and are used for deleting a library.

## Deleting a library

To delete a library from a port, use:

- In algebraic mode: PURGE(:port_number: lib_number)
- In RPN mode: : port_number : lib_number PURGE

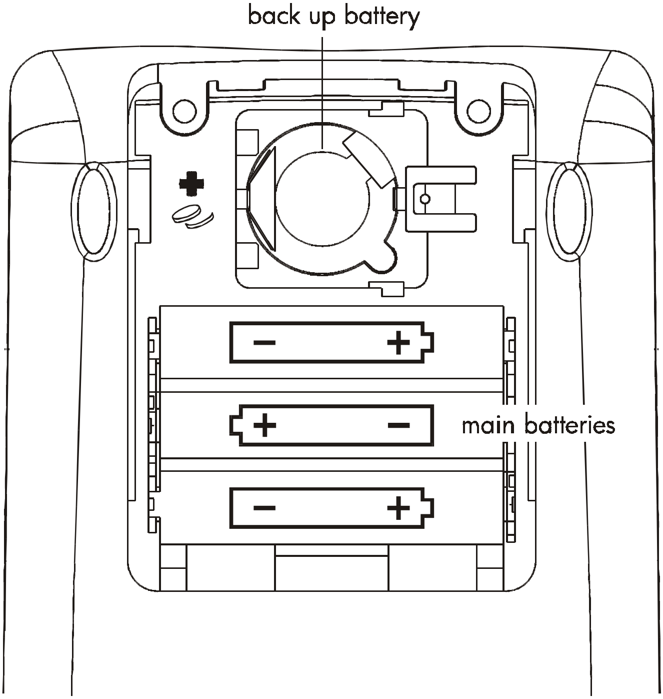
Where lib_number is the library number described above.

## Creating libraries

A library can be written in Assembler language, in System RPL language, or by using a library-creating library such as LBMKR. The latter program is available online (see for example, http://www.hpcalc.org). The details of programming the calculator in Assembler language or in System RPL language are beyond the scope of this document. The user is invited to find additional information on the subject online.

# Backup battery

A CR2032 back up battery is included in the calculator to provide power backup to volatile memory when changing the main batteries. It is recommended that you replace this battery every 5 years. A screen message will indicate when this battery needs replacement. The diagram below shows the location of the backup battery in the top compartment at the back of the calculator.



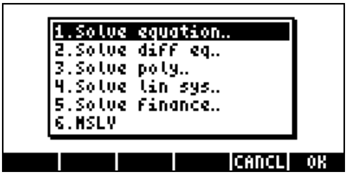
# Appendix A
# Using input forms

This example of setting time and date illustrates the use of input forms in the calculator. Some general rules:

- Use the arrow keys (◀▶▼▲) to move from one field to the next in the input form.
- Use any the CHOOS soft menu key to see the options available for any given field in the input form.
- Use the arrow keys (◀▶▼▲) to select the preferred option for a given field, and press the OK (F6) soft menu key to make the selection.
- In some instances, a check mark is required to select an option in an input form. In such case use the CHK soft menu key to toggle the check mark on and off.
- Press the CANCL soft menu key to close an input form and return to the stack display. You can also press the ENTER key or the ON key to close the input form.

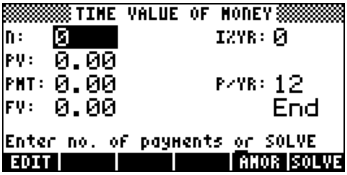
## Example - Using input forms in the NUM.SLV menu

Before discussing these items in detail we will present some of the characteristics of the input forms by using input forms from the financial calculation application in the numerical solver. Launch the numerical solver by using ↱ NUM.SLV (associated with the 7 key). . This produces a choose box that includes the following options:



1.Solve equation..
2.Solve diff eq..
3.Solve poly..
4.Solve lin sys..
5.Solve finance..
6.MSLV

To get started with financial calculations use the down arrow key (▼) to select item *5. Solve finance.* Press OK, to launch the application. The resulting screen is an input form with input fields for a number of variables (n, I%YR, PV, PMT, FV).

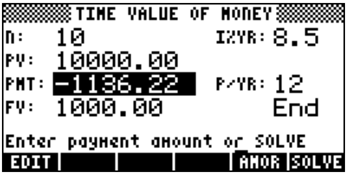


```
TIME VALUE OF MONEY
n:   0              I%YR: 0
PV:  0.00
PMT: 0.00           P/YR: 12
FV:  0.00                 End
Enter no. of payments or SOLVE
```

In this particular case we can give values to all but one of the variables, say, n = 10, I%YR = 8.5, PV = 10000, FV = 1000, and solve for variable PMT (the meaning of these variables will be presented later). Try the following:

|  |  |
|---|---|
| 10 OK | Enter n = 10 |
| 8.5 OK | Enter I%YR = 8.5 |
| 10000 OK | Enter PV = 10000 |
| ▼ 1000 OK | Enter FV = 1000 |
| ▲ ◀ SOLVE | Select and solve for PMT |

The resulting screen is:

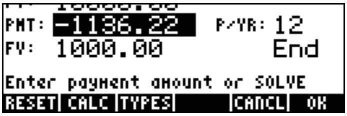


```
TIME VALUE OF MONEY
n:   10             I%YR: 8.5
PV:  10000.00
PMT: -1136.22       P/YR: 12
FV:  1000.00              End
Enter payment amount or SOLVE
```

In this input form you will notice the following soft menu key labels:

EDIT Press to edit highlighted field
AMOR Amortization menu - option specific to this application
SOLVE Press to solve for highlighted field

Pressing NXT we see the following soft menu key labels:

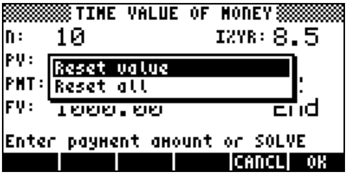


```
PMT: -1136.22       P/YR: 12
FV:  1000.00              End
Enter payment amount or SOLVE
```

RESET Reset fields to default values

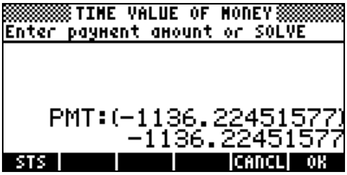
CALC Press to access the stack for calculations
TYPES Press to determine the type of object in highlighted field
CANCL Cancel operation
OK Accept entry

If you press RESET you will be asked to select between the two options:



```
TIME VALUE OF MONEY
n:  10          I%YR: 8.5
PV:   Reset value
PMT:  Reset all
FV:   1000.00        End
Enter payment amount or SOLVE
```

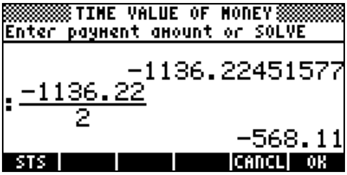
If you select *Reset value* only the highlighted value will be reset to the default value. If, instead, you select *Rest all*, all the fields will be reset to their default values (typically, 0). At this point you can accept your choice (press OK), or cancel the operation (press CANCL). Press CANCL in this instance. Press CALC to access the stack. The resulting screen is the following:



```
TIME VALUE OF MONEY
Enter payment amount or SOLVE

   PMT:(-1136.22451577)
         -1136.22451577
```

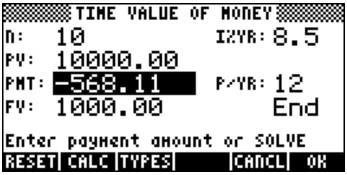
At this point, you have access to the stack, and the value last highlighted in the input form is provided for you. Suppose that you want to halve this value. The following screen follows in ALG mode after entering

1136.22/2:



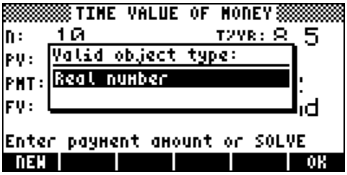
```
TIME VALUE OF MONEY
Enter payment amount or SOLVE
          -1136.22451577
:-1136.22
    2
                 -568.11
```

(In RPN mode, we would have used 1136.22 ENTER 2 ENTER ÷ ).
Press OK to enter this new value. The input form will now look like this:



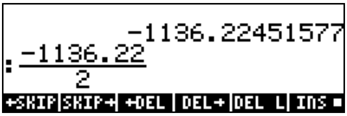
```
TIME VALUE OF MONEY
N:   10              I%YR: 8.5
PV:  10000.00
PMT: -568.11         P/YR: 12
FV:  1000.00               End
Enter payment amount or SOLVE
```

Press TYPES to see the type of data in the PMT field (the highlighted field). As a result, you get the following specification:



```
TIME VALUE OF MONEY
N:   10              I%YR: 8.5
PV:  Valid object type:
PMT: Real number
FV:
Enter payment amount or SOLVE
```

This indicates that the value in the PMT field must be a real number. Press OK to return to the input form, and press L to recover the first menu. Next, press the ENTER key or the ON key to return to the stack. In this instance, the following values will be shown:

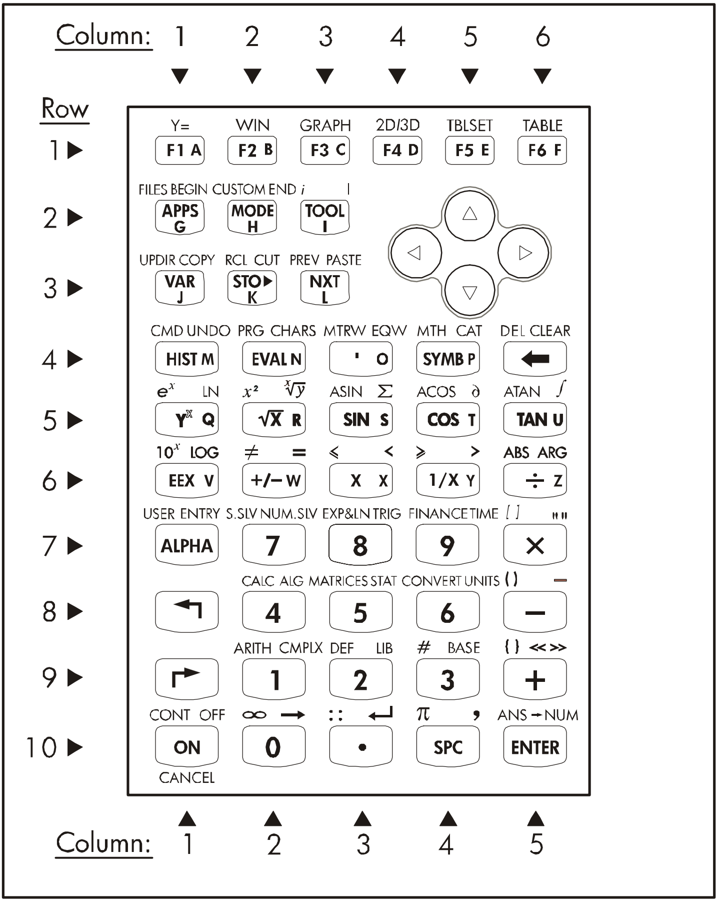


```
            -1136.22451577
: -1136.22
  --------
     2
```

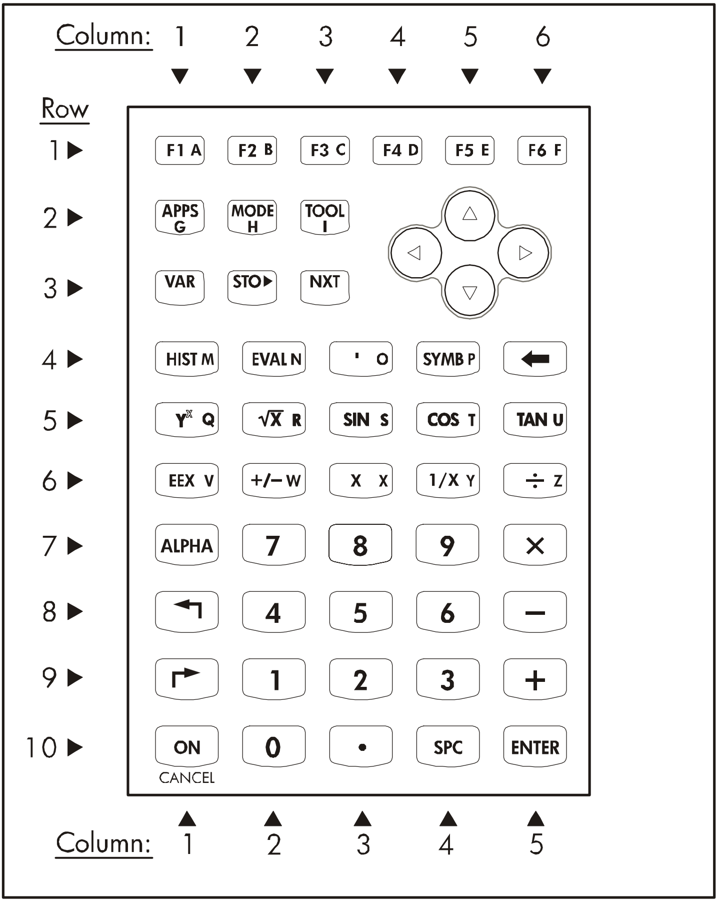
The top result is the value that was solved for PMT in the first part of the exercise. The second value is the calculation we made to redefine the value of PMT.

# Appendix B
# The calculator's keyboard

The figure below shows a diagram of the calculator's keyboard with the numbering of its rows and columns.


Column: 1 2 3 4 5 6
Row
1
2
3
4
5
6
7
8
9
10
Y= WIN GRAPH 2D/3D TBLSET TABLE
F1 A
F2 B
F3 C
F4 D
F5 E
F6 F
FILES BEGIN CUSTOM END i I
APPS G
MODE H
TOOL I
UPDIR COPY RCL CUT PREV PASTE
VAR J
STO▶ K
NXT L
CMD UNDO PRG CHARS MTRW EQW MTH CAT DEL CLEAR
HIST M
EVAL N
' O
SYMB P
ATAN ∫
eˣ LN x² ˣ√y ASIN Σ ACOS ∂
Yˣ Q
√X R
SIN S
COS T
TAN U
10ˣ LOG ≠ = ≤ < ≥ > ABS ARG
EEX V
+/− W
X X
1/X Y
÷ Z
USER ENTRY S.SLV NUM.SLV EXP&LN TRIG FINANCE TIME [ ] " "
ALPHA
7
8
9
×
CALC ALG MATRICES STAT CONVERT UNITS ( ) _
4
5
6
−
ARITH CMPLX DEF LIB # BASE { } « »
1
2
3
+
CONT OFF ∞ → : : ↵ π , ANS →NUM
ON
0
•
SPC
ENTER
CANCEL
Column: 1 2 3 4 5

The figure shows 10 rows of keys combined with 3, 5, or 6 columns. Row 1 has 6 keys, rows 2 and 3 have 3 keys each, and rows 4 through 10 have 5 keys each. There are 4 arrow keys located on the right-hand side of the keyboard in the space occupied by rows 2 and 3. Each key has three, four, or five functions. The main key functions are shown in the figure below. To operate this main key functions simply press the corresponding key. We will refer to the keys by the row and column where they are located in the sketch above, thus, *key (10,1)* is the *ON* key.



**Main key functions in the calculator's keyboard**

# Main key functions

Keys F1 through F6 keys are associated with the soft menu options that appear at the bottom of the calculator's display. Thus, these keys will activate a variety of functions that change according to the active menu.

- The arrow keys, ▲▼◀▶, are used to move one character at a time in the direction of the key pressed (i.e., up, down, left, or right).
- The *APPS* function activates the applications menu.
- The *MODE* function activates the calculator's modes menu.
- The *TOOL* function activates a menu of tools useful for handling variables and getting help on the calculator.
- The *VAR* function shows the variables stored in the active directory, the *STO* function is used to store contents in variables.
- The *NXT* function is used to see additional soft menu options or variables in a directory.
- The *HIST* function allows you access to the algebraic-mode history, i.e., the collection of recent command entries in that mode.
- The *EVAL* key is used to evaluate algebraic and numeric expressions, the apostrophe key [ ' ] is used to enter a set of apostrophes for algebraic expressions.
- The *SYMB* activates the symbolic operations menu.
- The delete key ◀ is used to delete characters in a line.
- The $y^x$ key calculates the $x$ power of $y$.
- The $\sqrt{x}$ key calculates the square root of a number.
- The *SIN*, *COS*, and *TAN* keys calculate the sine, cosine, and tangent, respectively, of a number.
- The *EEX* key is used to enter power of tens (e.g., $5\times10^3$, is entered as 5 EEX 3, which is shown as *5E3*).
- The *+/-* key changes the sign of an entry, the *X* key enters the character X (upper case).
- The *1/x* key calculates the inverse of a number, the keys $+$, $-$, $\times$, and $\div$, are used for the fundamental arithmetic operations (addition, subtraction, multiplication, and division, respectively).
- The *ALPHA* key is combined with other keys to enter alphabetic characters.

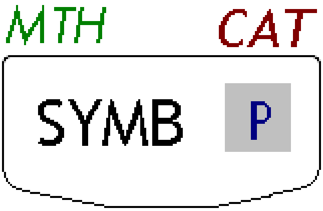
- The *left-shift* key ↰ and the *right-shift* key ↱ are combined with other keys to activate menus, enter characters, or calculate functions as described elsewhere.
- The *numerical keys* (*0* to *9*) are used to enter the digits of the decimal number system.
- There is a *decimal point key* (.) and a space key (*SPC*).
- The *ENTER* key is used to enter a number, expression, or function in the display or stack, and
- The *ON* key is used to turn the calculator on.

## Alternate key functions

The green left-shift key, *key (8,1)*, the red right-shift key, *key (9,1)*, and the blue ALPHA key, *key (7,1)*, can be combined with some of the other keys to activate the alternative functions shown in the keyboard. For example, the SYMB key, *key(4,4)*, has the following six functions associated with it:

| | |
|---|---|
| SYMB | Main function, to activate the SYMBolic menu |
| ↰ MTH | Left-shift function, to activate the MTH (Math) menu |
| ↱ CAT | Right-shift function, to activate the CATalog function |
| ALPHA P | ALPHA function, to enter the upper-case letter P |
| ALPHA ↰ P | ALPHA-Left-Shift function, to enter the lower-case letter p |
| ALPHA ↱ P | ALPHA-Right-Shift function, to enter the symbol P |

Of the six functions associated with the key only the first four are shown in the keyboard itself. This is the way that the key looks in the keyboard:



Notice that the color and the position of the labels in the key, namely, **SYMB**, *MTH*, *CAT* and **P**, indicate which is the main function (**SYMB**), and which of the other three functions is associated with the left-shift ↰ (*MTH*), right-shift ↱ (*CAT* ) , and ALPHA (**P**) keys.

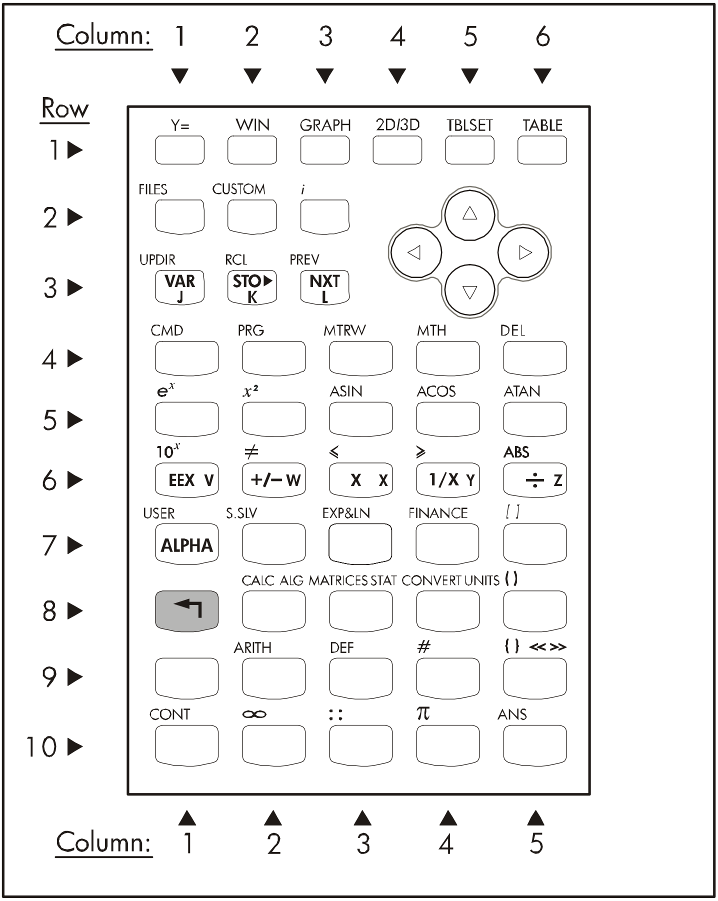
Diagrams showing the function or character resulting from combining the calculator keys with the left-shift ↰, right-shift ↱, ALPHA ALPHA, ALPHA-left-shift ALPHA↰, and ALPHA-right-shift ALPHA↱, are presented next. In these diagrams, the resulting character or function for each key combination is shown in white background. If the left-shift, right-shift or ALPHA keys are activated they are shown in a shaded background. Keys that do not get activated are shown in black background.

## Left-shift functions

The following sketch shows the functions, characters, or menus associated with the different calculator keys when the left-shift key ↰ is activated.

- The six left-shift functions associated with the F1 through F6 keys are associated with the setting up and production of graphics and tables. When using these functions in the calculator's *Algebraic* mode of operation, press the left-shift key ↰ first, and then any of the keys in Row 1. When using these functions in the calculator's ***RPN*** mode, you need to press the left-shift key ↰ simultaneously with the key in Row 1 of your choice. Function *Y=* is used to enter functions of the form *y=f(x)* for plotting, function *WIN* is used to set parameters of the plot window, function *GRAPH* is used to produce a graph, function *2D/3D* is used to select the type of graph to be produced, function *TBLSET* is used to set parameters for a table of values of a function, function *TABLE* is used to generate a table of values of a function,
- Function *FILE* activates the file browser in the calculator's memory.
- The *CUSTOM* function activates the custom menu options, the *i* key is used to enter the unit imaginary number *i* into the stack ($i^2 = -1$).
- The *UPDIR* function moves the memory location one level up in the calculator's file tree.
- The *RCL* function is used to recall values of variables.
- The *PREV* function shows the previous set of six menu options associated with the soft menu keys.

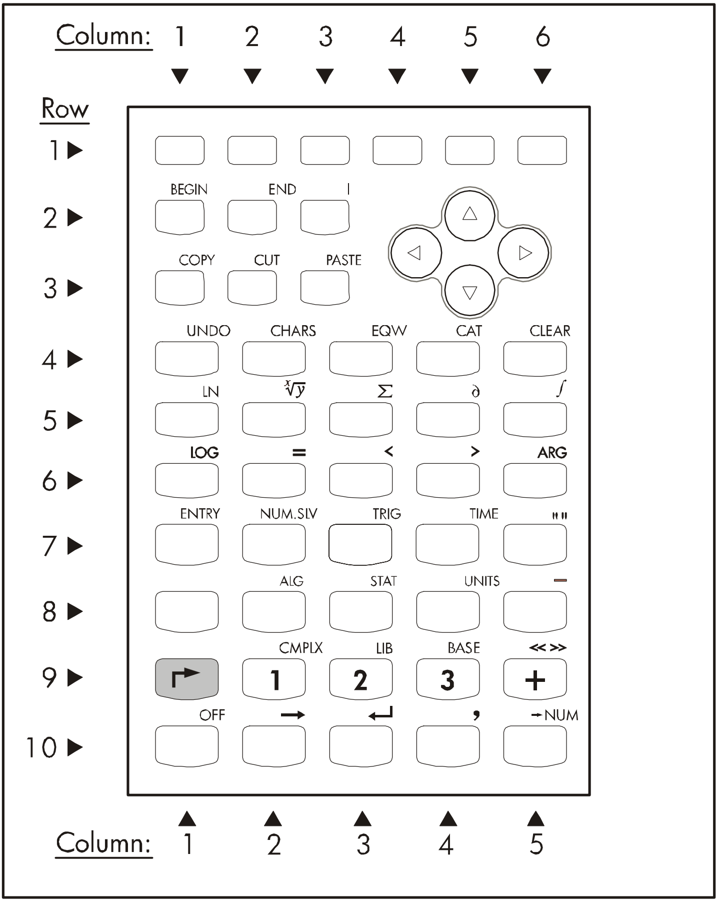
- The *CMD* function shows the most recent commands, the *PRG* function activates the programming menus, the *MTRW* function activates the matrix writer,



***Left-shift ⬑ functions of the calculator's keyboard***

- The *CMD* function shows the most recent commands.
- The *PRG* function activates the programming menus.
- The *MTRW* function activates the matrix writer.
- The *MTH* function activates a menu of mathematical function.
- The *DEL* key is used to delete variables.

- The $e^x$ key calculates the exponential function of *x*.
- The $x^2$ key calculates the square of *x* (this is referred to as the *SQ* function).
- The ASIN, ACOS, and ATAN functions calculate the arcsine, arccosine, and arctangent functions, respectively.
- The $10^x$ function calculates the anti-logarithm of *x*.
- The keys $\neq$, $\leq$, and $\geq$, are used for comparing real numbers.
- The *ABS* function calculates the absolute value of a real number, or the magnitude of a complex number or of a vector.
- The *USER* function activates the user-defined keyboard menu.
- The *S.SLV* function activates the symbolic solver menu.
- The *EXP&LN* function activates the menu for substituting expressions in terms of the exponential and natural logarithm functions.
- The *FINANCE* function activates a menu for financial calculations.
- The *CALC* function activates a menu of calculus functions.
- The *MATRICES* function activates a menu for creating and manipulation of matrices.
- The *CONVERT* function activates a menu for conversion of units and other expressions.
- The *ARITH* function activates a menu of arithmetic functions.
- The *DEF* key is used to define a simple function as a variable in the calculator menu.
- The *CONT* key is used to continue a calculator operation.
- The *ANS* key recalls the last result when the calculator is in Algebraic operation mode.
- The *[ ]*, *( )*, and *{ }* keys are used to enter brackets, parentheses, or braces.
- The *#* key is used to enter numbers in other than the active number base.
- The infinity key $\infty$ is used to enter the infinite symbol in an expression.
- The pi key $\pi$ is used to enter the value or symbol for $\pi$ (the ratio of the length of a circumference to its diameter).
- The arrow keys, when combined with the left-shift key, move the cursor to the first character in the direction of the key pressed.



***Right-shift ⮣ functions of the calculator's keyboard***

## Right-shift functions

The sketch above shows the functions, characters, or menus associated with the different calculator keys when the right-shift key ⮣ is activated.

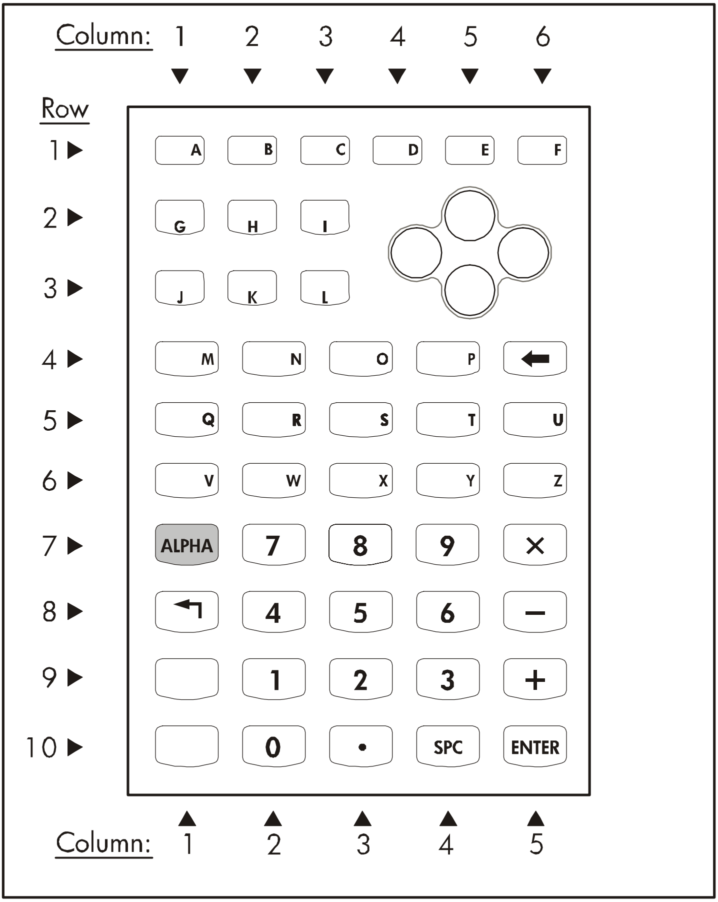
- The functions *BEGIN, END, COPY, CUT* and *PASTE* are used for editing purposes.
- The *UNDO* key is used to undo the last calculator operation.
- The *CHARS* function activates the special characters menu.
- The *EQW* function is used to start the Equation Writer.

- The *CAT* function is used to activate the command catalog.
- The *CLEAR* function clears the screen.
- The *LN* function calculates the natural logarithm.
- The $\sqrt[x]{y}$ function calculates the *x* – th root of *y*.
- The $\Sigma$ function is used to enter summations (or the upper case Greek letter sigma).
- The $\partial$ function is used to calculate derivatives.
- The $\int$ function is used to calculate integrals.
- The *LOG* function calculates the logarithm of base 10.
- The *ARG* function calculates the argument of a complex number.
- The *ENTRY* function is used to change entry mode in editing.
- The *NUM.SLV* function launches the NUMerical SOLver menu.
- The *TRIG* function activates the trigonometric substitution menu.
- The *TIME* function activates the time menu.
- The *ALG* function activates the algebra menu.
- The STAT function activates the statistical operations menu.
- The *UNITS* function activates the menu for units of measurement.
- The *CMPLX* function activates the complex number functions menu.
- The *LIB* function activates the library functions.
- The *BASE* function activates the numeric base conversion menu.
- The OFF key turns the calculator off, the *→NUM* key produces a numeric (or floating-point) value of an expression.
- The *" "* key enters a set of double-quotes used for entering text strings.
- The __ key enters an underscore.
- The << >> key enters the symbol for a program.
- The → key enters an arrow representing an input in a program.
- The ↵ key enters a return character in programs or text strings.
- The comma (,) key enters a comma.
- The arrow keys, when combined with the right-shift key, move the cursor to the farthest character in the direction of the key pressed.

## ALPHA characters

The following sketch shows the characters associated with the different calculator keys when the ALPHA ALPHA is activated. Notice that the ALPHA function is used mainly to enter the upper-case letters of the English alphabet

(*A* through *Z*). The numbers, mathematical symbols (-, +), decimal point (.), and the space (*SPC*) are the same as the main functions of these keys. The ALPHA function produces an asterisk (*) when combined with the times key, i.e., ALPHA ×.

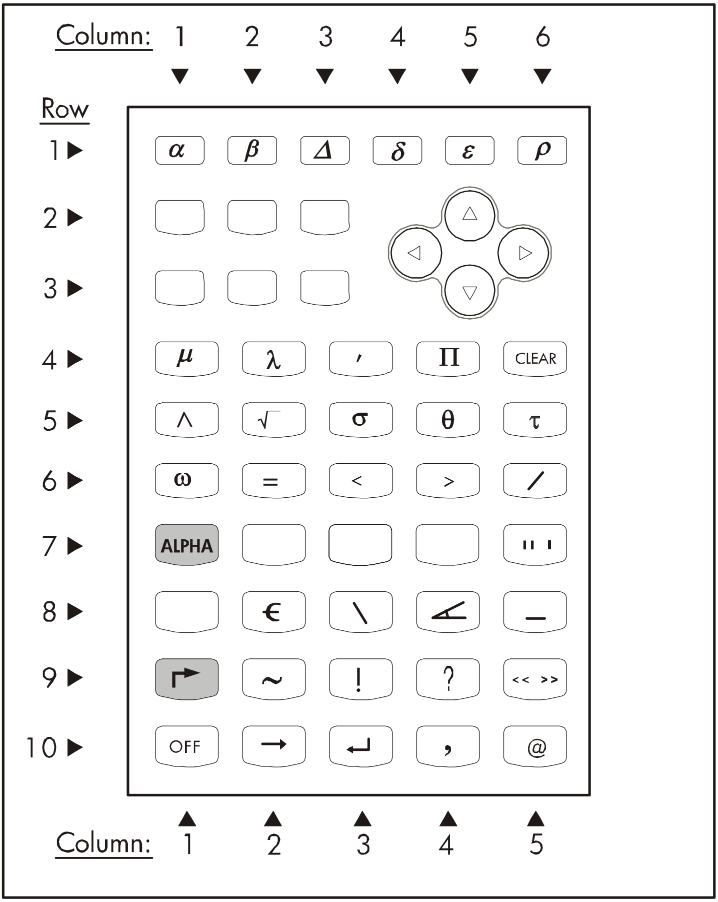


**Alpha ALPHA functions of the calculator's keyboard**

## Alpha-left-shift characters

The following sketch shows the characters associated with the different calculator keys when the ALPHA ALPHA is combined with the left-shift key ↰.

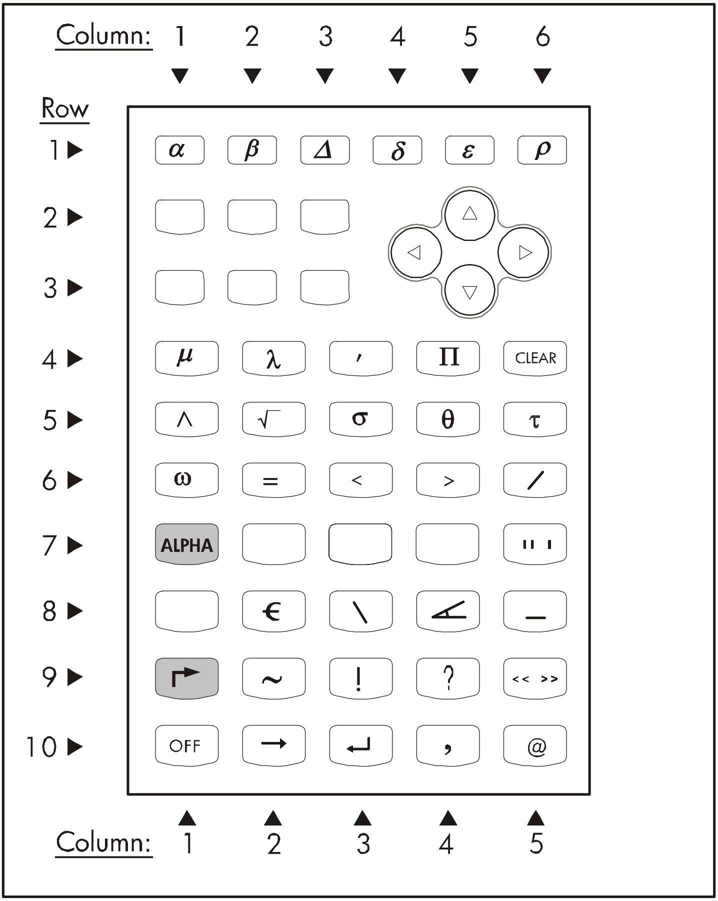
Notice that the ALPHA ← combination is used mainly to enter the lower-case letters of the English alphabet (*A* through *Z*). The numbers, mathematical symbols (-, +, ×), decimal point (.), and the space (*SPC*) are the same as the main functions of these keys. The ENTER and CONT keys also work as their main function even when the ALPHA ← combination is used.



**Alpha ALPHA ← functions of the calculator's keyboard**

## Alpha-right-shift characters

The following sketch shows the characters associated with the different calculator keys when the ALPHA ALPHA is combined with the right-shift key ↱.



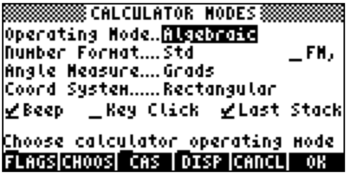
**Alpha ALPHA ↱ functions of the calculator's keyboard**

Notice that the ALPHA ↱ combination is used mainly to enter a number of special characters from into the calculator stack. The CLEAR, OFF, →, ↵, comma (,), key enters and OFF keys also work as their main function even when the ALPHA ↱ combination is used. The special characters generated by the ALPHA ↱ combination include Greek letters (α, β, Δ, δ, ε, ρ, μ, λ, σ, θ, τ, ω, and Π), other characters generated by the ALPHA ↱ combination are |, ‘, ^, =, <, >, /, “, \, __, ~, !, ?, <<>>, and @.

# Appendix C
# CAS settings

CAS stands for *C*omputer *A*lgebraic *S*ystem. This is the mathematical core of the calculator where the symbolic mathematical operations and functions are programmed. The CAS offers a number of settings can be adjusted according to the type of operation of interest. To see the optional CAS settings use the following:

- Press the MODE button to activate the CALCULATOR MODES input form.



```
CALCULATOR MODES
Operating Mode..Algebraic
Number Format....Std          _FM,
Angle Measure....Grads
Coord System......Rectangular
✓Beep   _Key Click   ✓Last Stack

Choose calculator operating mode
FLAGS|CHOOS| CAS |DISP |CANCL| OK
```

At the bottom of the display you will find the following soft menu key options:

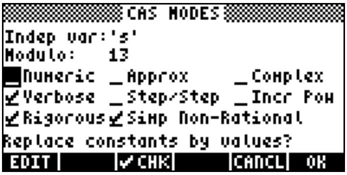
| | |
|---|---|
| FLAGS | Provides menus for manipulating calculator flags (*) |
| CHOOS | Lets the user chose options in the different fields in the form |
| CAS | Provides an input form to change CAS settings |
| DISP | Provides an input form to change display settings |
| CANCL | Closes this input form and returns to normal display |
| OK | Use this key to accept settings |

(*) Flags are variables in the calculator, referred to by numbers, which can be "set" and "unset" to change certain calculator operating options.

Pressing the NXT key shows the remaining options in the CALCULATOR MODES input form:

| | |
|---|---|
| RESET | Allows the user to reset a highlighted option |
| CANCL | Closes this input form and returns to normal display |
| OK | Use this key to accept settings |

- To recover the original menu in the CALCULATOR MODES input box, press the NXT key. Of interest at this point is the changing of the CAS settings. This is accomplished by pressing the CAS soft menu key. The default values of the CAS setting are shown below:



```
CAS MODES
Indep var:'s'
Modulo:   13
_Numeric _Approx    _Complex
✓Verbose _Step/Step _Incr Pow
✓Rigorous✓Simp Non-Rational
Replace constants by values?
EDIT |    |✓CHK|    |CANCL| OK
```

- To navigate through the many options in the CAS MODES input form, use the arrow keys: ◀ ▶ ▼ ▲ .

- To select or deselect any of the settings shown above, select the underline before the option of interest, and toggle the CHK soft menu key until the right setting is achieved. When an option is selected, a check mark will be shown in the underline (e.g., the *Rigorous* and *Simp Non-Rational* options above). Unselected options will show no check mark in the underline preceding the option of interest (e.g., the *_Numeric, _Approx, _Complex, _Verbose, _Step/Step, _Incr Pow* options above).

- After having selected and unselected all the options that you want in the CAS MODES input form, press the OK soft menu key. This will take you back to the CALCULATOR MODES input form. To return to normal calculator display at this point, press the OK soft menu key once more.

## Selecting the independent variable

Many of the functions provided by the CAS use a pre-determined independent variable. By default, such variable is chosen to be the letter *X* (upper case) as shown in the CAS MODES input box above. However, the user can change this variable to any other letter or combination of letters and numbers (a variable name must start with a letter) by editing the *Indep var* field in the CAS MODES input box.

A variable called VX exists in the calculator's {HOME CASDIR} directory that takes, by default, the value of 'X'. This is the name of the preferred independent variable for algebraic and calculus applications. For that reason, most examples in this Chapter use X as the unknown variable. If you use other independent variable names, for example, with function HORNER, the CAS will not work properly.

The variable VX is a permanent inhabitant of the {HOME CASDIR} directory. There are other CAS variables in the {HOME CASDIR}, e.g., REALASSUME (REALA), MODULO (MODUL), CASINFO (CASIN), etc.

You can change the value of VX by storing a new algebraic name in it, e.g., 'x', 'y', 'm', etc. Preferably, keep 'X' as your VX variable for the examples in this manual.

Also, avoid using the variable VX in your programs or equations, so as to not get it confused with the CAS' VX. If you need to refer to the x-component of velocity, for example, you can use vx or Vx.

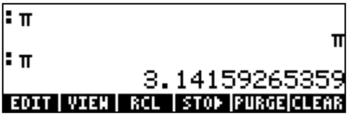
## Selecting the modulus

The *Modulo* option of the CAS MODES input box represents a number (default value = *13*) used in modular arithmetic. More details about modular arithmetic are presented elsewhere.

## Numeric vs. symbolic CAS mode

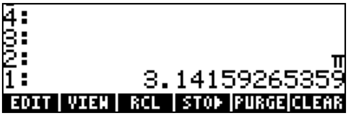
When the *Numeric* CAS mode is selected, certain constants pre-defined in the calculator are displayed in their full floating-point value. By default, the _*Numeric* option is unselected, meaning that those pre-defined constants will be displayed as their symbol, rather than their value, in the calculator display.

The following screen shows the values of the constant $\pi$ (the ratio of the length of the circumference to its diameter) in symbolic format followed by the numeric, or floating-point, format. This example corresponds to the Algebraic operating mode.



```
:π
                    π
:π
        3.14159265359
```

The same example, corresponding to the RPN operating mode, is shown next:

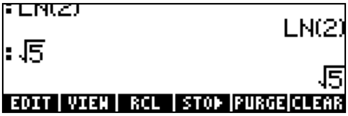


```
4:
3:
2:                  π
1:      3.14159265359
```

## Approximate vs. Exact CAS mode

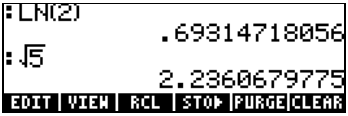
When the *_Approx* is selected, symbolic operations (e.g., definite integrals, square roots, etc.), will be calculated numerically. When the *_Approx* is unselected (Exact mode is active), symbolic operations will be calculated as closed-form algebraic expressions, whenever possible.

The following screen shows a couple of symbolic expressions entered with an active exact mode in Algebraic operating mode:



```
:LN(2)
                LN(2)
:√5
                   √5
```

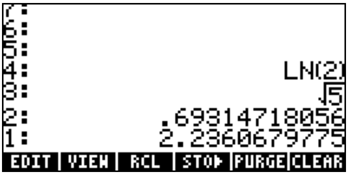
In Algebraic mode, the object entered by the user is shown in the left-hand side of the screen, followed immediately by a result in the right-hand side of the screen. The results shown above show the symbolic expressions for *ln(2)*, i.e., the natural logarithm of 2, and $\sqrt{5}$, i.e., the square root of 5. If the *_Numeric* CAS option is selected, the corresponding results for these operations are as follows:



```
:LN(2)
         .69314718056
:√5
         2.2360679775
```

The keystrokes necessary for entering these values in Algebraic mode are the following: ↱ LN 2 ENTER √x 5 ENTER

The same calculations can be produced in RPN mode. Stack levels *3:* and *4:* show the case of Exact CAS setting (i.e., the *_Numeric* CAS option is unselected), while stack levels *1:* and *2:* show the case in which the *Numeric CAS* option is selected.



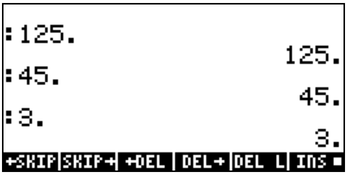
```
7:
6:
5:
4:                LN(2)
3:                   √5
2:         .69314718056
1:         2.2360679775
```

The required keystrokes are: 2 ↱ LN 5 √x

A keyboard short cut to toggle between APPROX and EXACT mode is by holding the right-shift key and pressing the ENTER key simultaneously, i.e., ↱ (hold) ENTER.

## Real numbers vs. integer numbers

CAS operations utilize integer numbers in order to keep full precision in the calculations. Real numbers are stored in the form of a mantissa and an exponent, and have limited precision. In APPROX mode, however, whenever you enter an integer number, it is automatically transformed into a real number, as illustrated next:



```
:125.
                 125.
:45.
                  45.
:3.
                   3.
```

Whenever the calculator lists an integer value followed by a decimal dot, it is indicating that the integer number has been converted to a real representation. This will indicate that the number was entered while the CAS was set to APPROX mode.

It is recommended that you select EXACT mode as default CAS mode, and change to APPROX mode if requested by the calculator in the performance of an operation.

For additional information on real and integer numbers, as well as other calculator's objects, refer to Chapter 2.

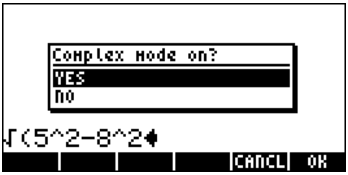
## Complex vs. Real CAS mode

A complex number is a number of the form *a+bi*, where *i*, defined by $i^2 = -1$ is the unit imaginary number (electrical engineers prefer to use the symbol *j*), and *a* and *b* are real numbers. For example, the number *2 + 3i* is a complex number. Additional information on operations with complex numbers are presented in Chapter 4 of this guide.

When the *_Complex* CAS option is selected, if an operation results in a complex number, then the result will be shown in the form *a+bi* or in the form of an ordered pair *(a,b)*. On the other hand, if the *_Complex* CAS option is unset (i.e., the Real CAS option is active), and an operation results in a complex number, you will be asked to switch to Complex mode. If you decline, the calculator will report an error.

Please notice that, in COMPLEX mode the CAS is able to perform a wider range of operations than in REAL mode, but it will also be considerably slower. Thus, it is recommended that you use the REAL mode as default mode and switch to COMPLEX if requested by the calculator in the performance of an operation.

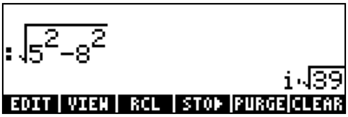
The following example shows the calculation of the quantity $\sqrt{5^2 - 8^2}$ using the Algebraic operating mode, first with the Real CAS option selected. In this case, you are asked if you want to change the mode to Complex:



Complex mode on?

```
√(5^2-8^2◆
```

If you press the OK soft menu key (), then the _Complex option is forced, and the result is the following:

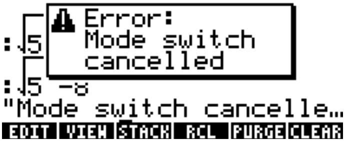


```
:√(5²-8²)
                    i·√39
```

The keystrokes used above are the following:

√x ← ()  5 ← yˣ 2 + 8 ← yˣ 2 ENTER

When asked to change to COMPLEX mode, use: F6 . If you decide not to accept the change to COMPLEX mode, you get the following error message:



```
Error:
Mode switch
cancelled
:√5
:√5 -8
"Mode switch cancelle…
```

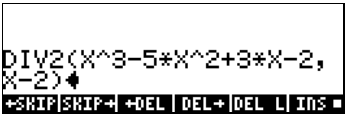
## Verbose vs. non-verbose CAS mode

When the _*Verbose* CAS option is selected, certain calculus applications are provided with comment lines in the main display. If the _*Verbose* CAS option is not selected, then those calculus applications will show no comment lines. The comment lines will appear momentarily in the top lines of the display while the operation is being calculated.

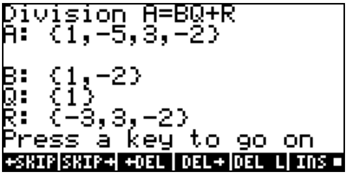
## Step-by-step CAS mode

When the _*Step/step* CAS option is selected, certain operations will be shown step at a time in the display. If the _*Step/step* CAS option is not selected, then intermediate steps will not be shown.

For example, having selected the Step/step option, the following screens show the step-by-step division of two polynomials, namely, $(X^3-5X^2+3X-2)/(X-2)$. This is accomplished by using function DIV2 as shown below. Press ENTER to show the first step:



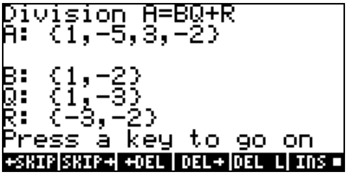
```
DIV2(X^3-5*X^2+3*X-2,
X-2)◆
```



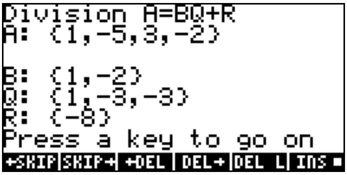
```
Division A=BQ+R
A: {1,-5,3,-2}

B: {1,-2}
Q: {1}
R: {-3,3,-2}
Press a key to go on
```

The screen inform us that the calculator is operating a division of polynomials A/B, so that A = BQ + R, where Q = quotient, and R = remainder. For the case under consideration, $A = X^3-5X^2+3X-2$, and $B = X-2$. These polynomials are represented in the screen by lists of their coefficients. For example, the expression A: {1,-5,3,-2} represents the polynomial $A = X^3-5X^2+3X-2$, B:{1,-2} represents the polynomial B = X-2, Q: {1} represents the polynomial Q = X, and R:{-3,3,-2} represents the polynomial $R = -3X^2+3X-2$.

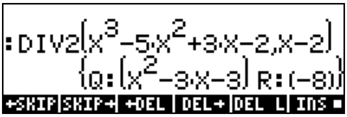
At this point, press, for example, the ENTER key. Continue pressing ENTER the key to produce additional steps:



```
Division A=BQ+R
A: {1,-5,3,-2}

B: {1,-2}
Q: {1,-3}
R: {-3,-2}
Press a key to go on
```



```
Division A=BQ+R
A: {1,-5,3,-2}

B: {1,-2}
Q: {1,-3,-3}
R: {-8}
Press a key to go on
```



$$:\mathrm{DIV2}(X^3-5\cdot X^2+3\cdot X-2,X-2)$$
$$\{Q:(X^2-3\cdot X-3)\ R:(-8)\}$$

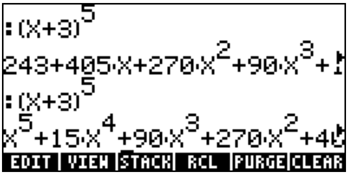
Thus, the intermediate steps shown represent the coefficients of the quotient and residual of the step-by-step synthetic division as would have been performed by hand, i.e.,

$$\frac{X^3-5X^2+3X-2}{X-2} = X^2 + \frac{-3X^2+3X-2}{X-2} =$$

$$X^2 - 3X + \frac{-3X-2}{X-2} = X^2 - 3X - 3X - \frac{8}{X-2}.$$

## Increasing-power CAS mode

When the *_Incr pow* CAS option is selected, polynomials will be listed so that the terms will have increasing powers of the independent variable. If the *_Incr pow* CAS option is not selected (default value) then polynomials will be listed so that the terms will have decreasing powers of the independent variable. An example is shown next in Algebraic mode:

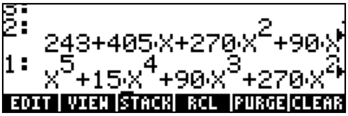


```
:(X+3)^5
243+405·X+270·X^2+90·X^3+
:(X+3)^5
X^5+15·X^4+90·X^3+270·X^2+4
```

In the first case, the polynomial $(X+3)^5$ is expanded in increasing order of the powers of *X*, while in the second case, the polynomial shows decreasing order of the powers of *X*. The keystrokes in both cases are the following:

↰ () X + 3 ▶ $y^x$ 5 ENTER

In the first case the *_Incr pow* option was selected, while in the second it was not selected. The same example, in RPN notation, is shown below:



```
3:
2:   243+405·X+270·X^2+90·X
1:   X^5+15·X^4+90·X^3+270·X^2
```

The same keystroke sequence was used to produce each of these results:

' ↰ () X + 3 ▶ $y^x$ 5 ENTER EVAL

## Rigorous CAS setting

When the *_Rigorous* CAS option is selected, the algebraic expression *|X|*, i.e., the absolute value, is not simplified to *X*. If the *_Rigorous* CAS option is not selected, the algebraic expression *|X|* is simplified to *X*.

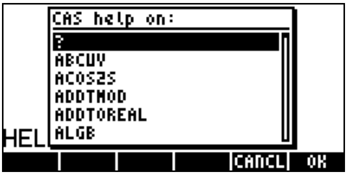
The CAS can solve a larger variety of problems if the rigorous mode is not set. However, the result, or the domain in which the result are applicable, might be more limited.

## Simplify non-rational CAS setting

When the *_Simp Non-Rational* CAS option is selected, non-rational expressions will be automatically simplified. On the other hand, if the *_Simp Non-Rational* CAS option is not selected, non-rational expressions will not be automatically simplified.

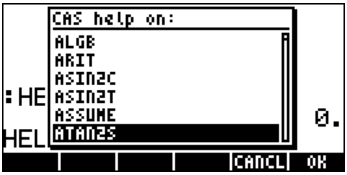
## Using the CAS HELP facility

Turn on the calculator, and press the TOOL key to activate the TOOL menu. Next, press the F2 soft menu key, followed by the ENTER key (the key in the lowest right corner of the keyboard), to activate the HELP facility. The display will look as follows:



At this point you will be provided with a list of all CAS commands in alphabetical order. You can use the down arrow key, ▼, to navigate through the list. To move upwards in the list use the up arrow key, ▲ . The arrow keys are located on the right-hand side of the keyboard between the first and fourth rows of keys.

Suppose that you want to find information on the command ATAN2S (ArcTANgent-to-Sine function). Press the down arrow key, ▼, until the command ATAN2S is highlighted in the list:



CAS help on:
ALGB
ARIT
ASIN2C
ASIN2T
ASSUME
ATAN2S

:HE
HEL
0.

CANCL OK

Notice that, in this instance, soft menu keys F5 and F6 are the only one with associated commands, namely:

CANCL F5 CANCeL the help facility
OK F6 OK to activate help facility for the selected command

If you press the CANCL F5 key, the HELP facility is skipped, and the calculator returns to normal display.

To see the effect of using OK in the HELP facility, let's repeat the steps used above from to the selection of the command ATAN2S in the list of CAS commands: HELP F2 ENTER ▼ ▼ …(10 times)

Then, press the OK F6 key to obtain information about the command ATAN2S.

The help facility indicates that the command, or function, ATAN2S replaces the value of *atan(x)*, the arc tangent of a value *x*, by its equivalent in terms of the function *asin* (arcsine), i.e.,

The fourth and fifth lines in the display provide an example of application of the function ATAN2S. Line four, namely, ATAN2S(ATAN(X)), is the statement of the operation to be performed, while line five, namely, ASIN(X/√(X^2+1)), is the result.

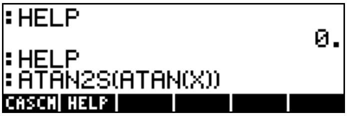
The bottom line in the display, starting with the particle *See:*, is a reference line listing other CAS commands related to the command ATAN2S.

Notice that there are six commands associated with the soft menu keys in this case (you can check that there are only six commands because pressing the

NXT produces no additional menu items). The soft menu key commands are the following:

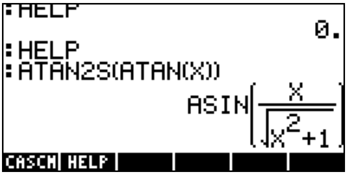
| | | |
|---|---|---|
| EXIT | F1 | EXIT the help facility |
| ECHO | F2 | Copy the example command to the stack and exit |
| SEE1 | F3 | See the first link (if any) in the list of references |
| SEE2 | F4 | See the second link (if any) of the list of references |
| SEE3 | F5 | See the third link (if any) of the list of references |
| MAIN | F6 | Return to the MAIN command list in the help facility |

In this case we want to ECHO the example into the stack by pressing ECHO F2 . The resulting display is the following:



```
:HELP
                        0.
:HELP
:ATAN2S(ATAN(X))
```

There are now four lines of the display occupied with output. The first two lines from the top correspond to the first exercise with the HELP facility in which we cancel the request for help. The third line from the top shows the most recent call to the HELP facility, while the last line shows the ECHO of the example command. To activate the command press the ENTER key. The result is:



```
:HELP
                        0.
:HELP
:ATAN2S(ATAN(X))
```

$$\mathrm{ASIN}\left(\frac{X}{\sqrt{X^2+1}}\right)$$

Notice that, as new lines of output are produced, the display (or stack) pushes the existing lines upwards and fills the bottom of the screen with more output.

The HELP facility, described in this section, will be very useful to refer to the definition of the many CAS commands available in the calculator. Each entry in the CAS help facility, whenever appropriate, will have an example of application of the command, as well as references as shown in this example.

To navigate quickly to a particular command in the help facility list without having to use the arrow keys all the time, we can use a shortcut consisting of typing the first letter in the command's name. Suppose that we want to find information on the command IBP (Integration By Parts), once the help facility list is available, use the ALPHA key (first key in the fourth row from the bottom of the keyboard) followed by the key for the letter i (the same as the key TOOL) , i.e., ALPHA I . This will take you automatically to the first command that starts with an *i*, namely, IBASIS. Then, you can use the down arrow key ▼ , twice, to find the command IBP. Pressing the OK F6 key, we activate the help facility for this command. Press MAIN F6 to recover the main list of commands, or EXIT F1 to exit the facility.

### References for non-CAS commands

The help facility contains entries for all the commands developed for the CAS (Computer Algebraic System). There is a large number of other functions and commands that were originally developed for the HP 48G series calculators that are not included in the help facility. Good references for those commands are the *HP 48G Series user's guide* (HP Part No. 00048-90126) and the *HP 48G Series Advanced User's Reference Manual* (HP Part No. 00048-90136) both published by Hewlett-Packard Company, Corvallis, Oregon, in 1993.

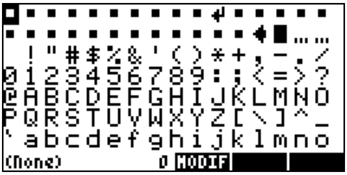
## CAS End User Term and Conditions

Use of the CAS Software requires from the user an appropriate mathematical knowledge. There is no warranty for the CAS Software, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holder provides the CAS Software "As Is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the CAS Software is with you. Should the CAS Software prove defective, you assume the cost of all necessary servicing, repair or correction.

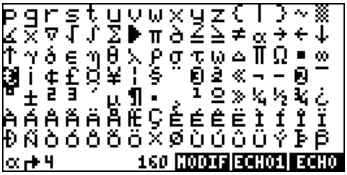
In no event unless required by applicable law will any copyright holder be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the CAS Software (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the CAS Software to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages. If required by applicable law the maximum amount payable for damages by the copyright holder shall not exceed the royalty amount paid by Hewlett-Packard to the copyright holder for the CAS Software.

# Appendix D
# Additional character set

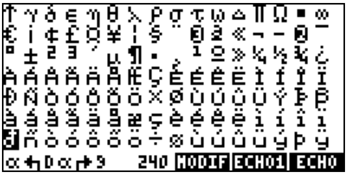
While you can use any of the upper-case and lower-case English letter from the keyboard, there are 255 characters usable in the calculator. Including special characters like θ, λ, etc., that that can be used in algebraic expressions. To access these characters we use the keystroke combination ↱ CHARS (associated with the EVAL key). The result is the following screen:



By using the arrow keys, ◀ ▶ ▼ ▲, we can navigate through the collection of characters. For example, moving downwards in the screen produces more characters in the display:



Moving farther down, we see these characters:



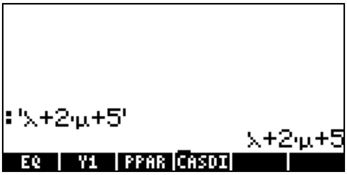
There will be one character highlighted at all times. The lower line in the display will show the short cut for the highlighted character, as well as the ASCII character code (e.g., see the screen above: the short cut is α←Dα→9, i.e., ALPHA ↰ D ALPHA ↱ 9 , and the code is *240*). The display also shows three functions associated with the soft menu keys, f4, f5, and f6. These functions are:

MODIF: Opens a graphics screen where the user can modify highlighted character. Use this option carefully, since it will alter the modified character up to the next reset of the calculator. (Imagine the effect of changing the graphic of the character 1 to look like a 2!).
ECHO1: Copies the highlighted character to the command line or equation writer (EQW) and exits the character set screen (i.e., echoes a single character to the stack).
ECHO: Copies the highlighted character to the command line or equation writer (EQW), but the cursor remains in the character set screen to allow the user to select additional characters (i.e., echoes a string of characters to the stack). To exit the character set screen press `.

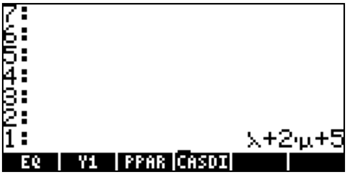
For example, suppose you have to type the expression: $\lambda^2 + 2\mu + 5$

Here is a suggested approach, using the stack in either Algebraic or RPN mode:

Use the keystrokes: ' ↱ CHARS to get to the characters screen. Next, use the arrow keys to highlight the character λ. Press ECHO1 (i.e., the F5 key), and continue with the keystrokes: + 2 × ↱ CHARS . Next, use the arrow keys to highlight the character μ. Press ECHO1 (i.e., the F5 key), and finish the expression with the keystrokes: + 5 ENTER . Here is the result of this exercise in Algebraic and RPN modes, respectively:



```
:'λ+2·μ+5'
                λ+2·μ+5
```



```
7:
6:
5:
4:
3:
2:
1:              λ+2·μ+5
```

Following, we list some of the most common ALPHA ↱ keystroke combinations:

## Greek letters

| | | |
|---|---|---|
| α | (alpha) | ALPHA ↱ A |
| β | (beta) | ALPHA ↱ B |
| δ | (delta) | ALPHA ↱ D |
| ε | (epsilon) | ALPHA ↱ E |
| θ | (theta) | ALPHA ↱ T |
| λ | (lambda) | ALPHA ↱ N |
| μ | (mu) | ALPHA ↱ M |
| ρ | (rho) | ALPHA ↱ F |
| σ | (sigma) | ALPHA ↱ S |
| τ | (tau) | ALPHA ↱ U |
| ω | (omega) | ALPHA ↱ V |
| Δ | (upper-case delta) | ALPHA ↱ C |
| Π | (upper-case pi) | ALPHA ↱ P |

## Other characters

| | | |
|---|---|---|
| ~ | (tilde) | ALPHA ↱ 1 |
| ! | (factorial) | ALPHA ↱ 2 |
| ? | (question mark) ) | ALPHA ↱ 3 |
| \ | (backward slash) | ALPHA ↱ 5 |
| ∡ | (angle symbol) | ALPHA ↱ 6 |
| @ | (at) | ALPHA ↱ ENTER |

Some characters commonly used that do not have simple keystroke shortcuts are: $\bar{x}$ (x bar), γ (gamma), η (eta), Ω (upper-case omega). These characters can be "echoed" from the CHARS screen: ↱ CHARS .

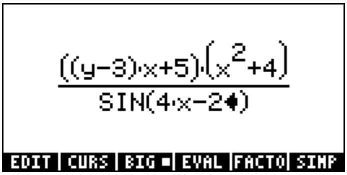
# Appendix E
# The Selection Tree in the Equation Writer

The expression tree is a diagram showing how the Equation Writer interprets an expression. The form of the expression tree is determined by a number of rules known as the hierarchy of operation. The rules are as follows:

1. Operations in parentheses are executed first, from the innermost to the outermost parentheses, and from left to right in the expression.
2. Arguments of functions are executed next, from left to right.
3. Functions are executed next, from left to right.
4. Powers of numbers are executed next, from left to right.
5. Multiplications and divisions are executed next, from left to right.
6. Additions and subtraction are executed last, from left to right.

Execution from left to right means that, if two operations of the same hierarchy, say two multiplications, exist in an expression, the first multiplication to the left will be executed before the second, and so on.

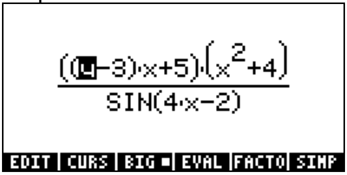
Consider, for example, the expression shown below in the equation writer:



$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2\blacklozenge)}$$

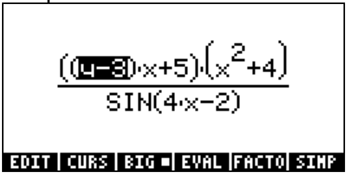
The insertion cursor (⬅) at this point is located to the right of the 2 in the argument of the SIN function in the denominator. Press the down arrow key ▼ to trigger the clear, editing cursor (□) around the 2 in the denominator. Next, press the left arrow key ◄ , continuously, until the clear, editing cursor is around the y in the first factor in the denominator. Then, press the upper-arrow key to activate the selection cursor (■) around the y. By pressing the upper arrow key ▲ , continuously, we can follow the expression tree that will take use from the y to the completion of the expression. Here is the sequence of operations highlighted by the upper arrow key▲ :

Step A1



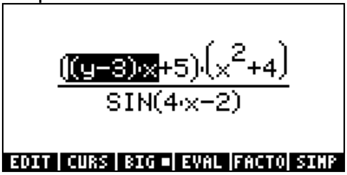
$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step A2



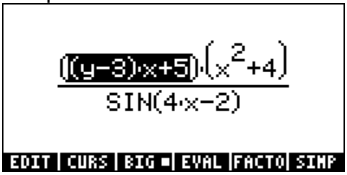
$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step A3



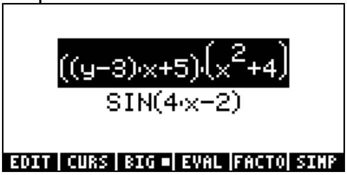
$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step A4



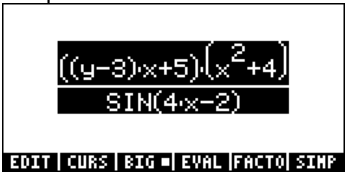
$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step A5



$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

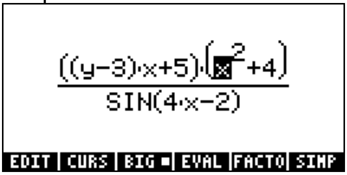
Step A6



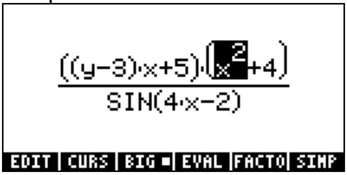
$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

We notice the application of the hierarchy-of-operation rules in this selection. First the y (Step A1). Then, y-3 (Step A2, parentheses). Then, (y-3)x (Step A3, multiplication). Then (y-3)x+5, (Step A4, addition). Then, ((y-3)x+5)(x$^2$+4) (Step A5, multiplication), and finally, ((y-3)x+5)(x$^2$+4)/SIN(4x-2) (Step A6, division). It is important to point out that the multiplication in Step A5 includes the first term, ((y-3)x+5) with a second term (x$^2$+4), which is already calculated. To see the steps in calculating these second term, press the down arrow key ▽, continuously, until the clear, editing cursor is triggered around the y, once more. Then, press the right arrow key until these cursor is over the x in the second term in the numerator. Then, press the upper-arrow key to select this x. The steps in the evaluation of the expression, starting from this point, are shown below:

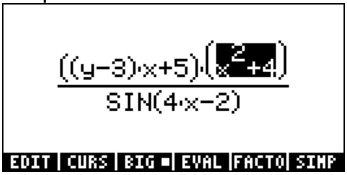
Step B1


$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

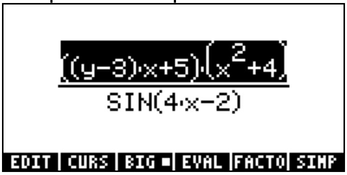
Step B2


$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

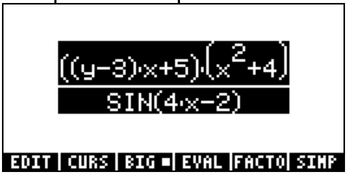
Step B3


$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step B4 = Step A5

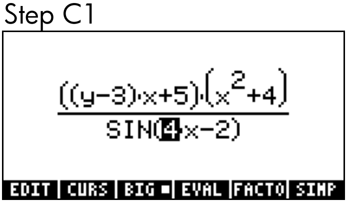

$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step B5 = Step A6

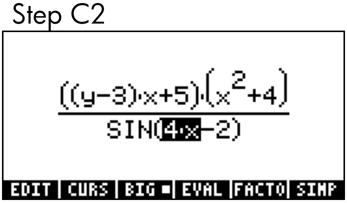

$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

We can also follow the evaluation of the expression starting from the 4 in the argument of the SIN function in the denominator. Press the down arrow key ▼, continuously, until the clear, editing cursor is triggered around the y, once more. Then, press the right arrow key until these cursor is over the 4 in the denominator. Then, press the upper-arrow key ▲ to select this 4. The steps in the evaluation of the expression, starting from this point, are shown below:

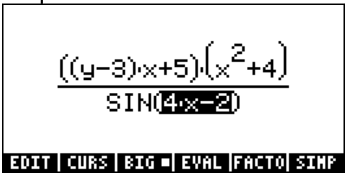
Step C1


$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step C2

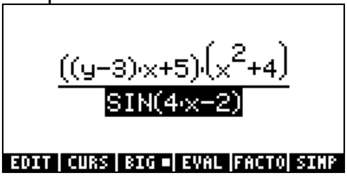

$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step C3



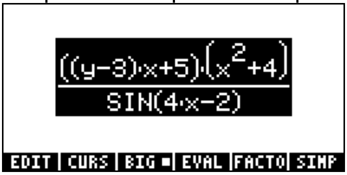
$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step C4



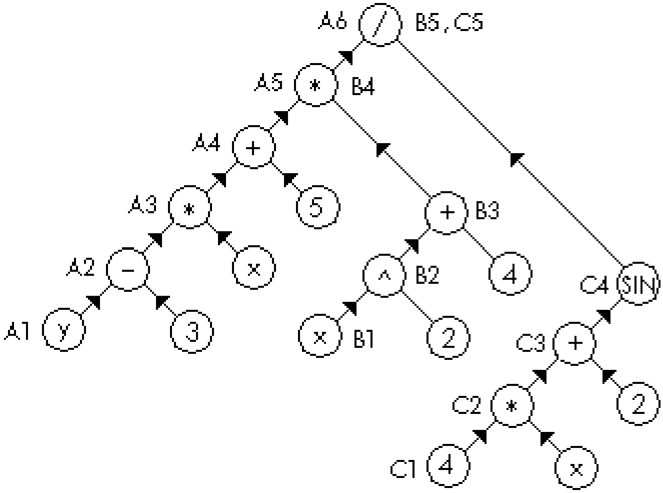
$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

Step C5 = Step B5 = Step A6



$$\frac{((y-3)\cdot x+5)\cdot(x^2+4)}{\mathrm{SIN}(4\cdot x-2)}$$

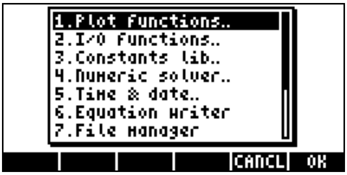
The expression tree for the expression presented above is shown next:

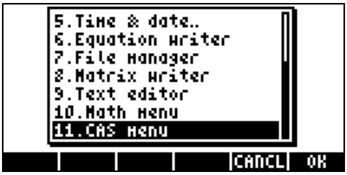


The steps in the evaluation of the three terms (A1 through A6, B1 through B5, and C1 through C5) are shown next to the circle containing numbers, variables, or operators.

# Appendix F
# The Applications (APPS) menu

The Applications (APPS) menu is available through the APPS key (first key in second row from the keyboard's top). The APPS key shows the following applications:

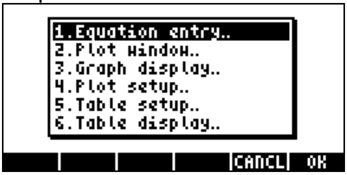




The different applications are described next.

## Plot functions..

Selecting option *1. Plot functions..* in the APPS will produce the following menu list of graph-related options:



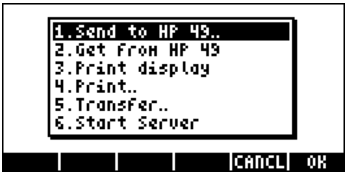
The six options shown are equivalent to the keystroke sequences listed below:

| | | | |
|---|---|---|---|
| Equation entry… | ↰ Y= | Plot window.. | ↰ WIN |
| Graph display.. | ↰ GRAPH | Plot setup.. | ↰ 2D/3D |
| Table setup.. | ↰ TBLSET | Table display.. | ↰ TABLE |

These applications are presented in detail in Chapter 12.

# I/O functions..

Selecting option *2. I/O functions..* in the APPS menu will produce the following menu list of input/output functions

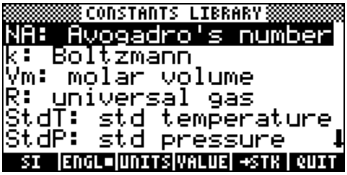


These applications are described next:

| | |
|---|---|
| Send to HP 49.. | Send data to another calculator |
| Get from HP 49 | Receive data from another calculator |
| Print display | Send screen to printer |
| Print.. | Print selected object from calculator |
| Transfer.. | Transfer data to other device |
| Start Server.. | Calculator set as a server for communication with computers |

# Constants lib..

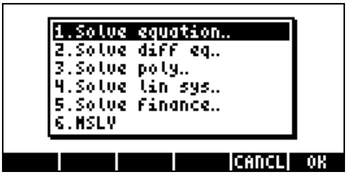
Selecting option *3. Constants lib..* in the APPS menu opens the Constant Library application that provides values of standard physical constants:



The Constants Library is discussed in detail in Chapter 3.

## Numeric solver..

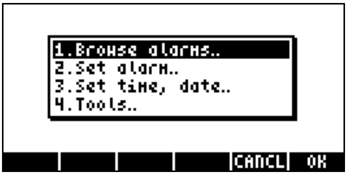
Selecting option *3. Constants lib..* in the APPS menu produces the numerical solver menu:



This operation is equivalent to the keystroke sequence ↱ NUM.SLV . The numerical solver menu is presented in detail in Chapters 6 and 7.

## Time & date..

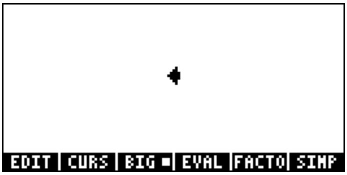
Selecting option *5.Time & date..* in the APPS menu produces the time and date menu:



This operation is equivalent to the keystroke sequence ↱ TIME . The time and date menu is presented in detail in Chapter 26.

## Equation writer..

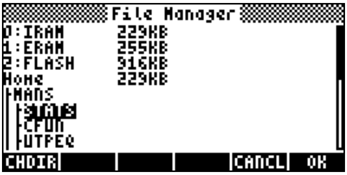
Selecting option *6.Equation writer..* in the APPS menu opens the equation writer:



This operation is equivalent to the keystroke sequence ↱ EQW . The equation writer is introduced in detail in Chapter 2. Examples that use the equation writer are available throughout this Guide.

## File manager..

Selecting option *7.File manager..* in the APPS menu launches the file manager application:

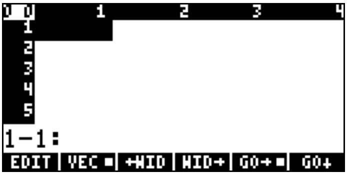


```
File Manager
0:IRAM      229KB
1:ERAM      255KB
2:FLASH     916KB
Home        229KB
├MANS
│ ├STATS
│ ├CFUN
│ ├UTPEQ
```

This operation is equivalent to the keystroke sequence ↰ FILES .The file manager is introduced in Chapter 2.

## Matrix writer..

Selecting option *8.Matrix writer..* in the APPS menu launches the matrix writer:

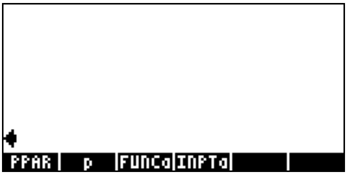


```
1-1:
```

This operation is equivalent to the keystroke sequence ↰ MTRW .The matrix writer is presented in detail in Chapter 10.

## Text editor..

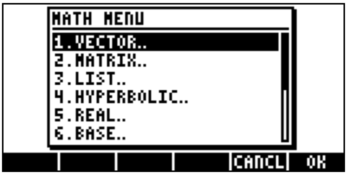
Selecting option *9.Text editor..* in the APPS menu launches the line text editor:

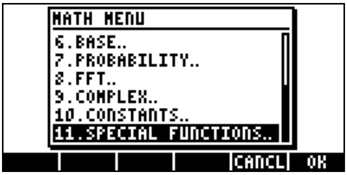


The text editor can be started in many cases by pressing the down-arrow key ▼ . If the object in the display is an algebraic object, pressing ▼ will most likely start the Equation Writer. The text editor is introduced in Chapter 2, and presented in detail in Appendix L.

## Math menu ..

Selecting option *10.Math menu..* in the APPS menu produces the MTH (mathematics) menu:

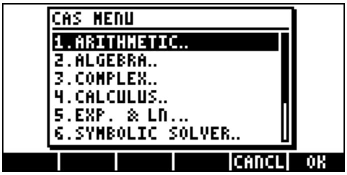


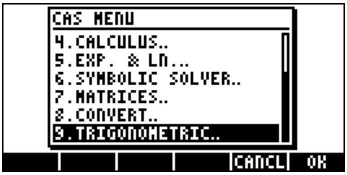


This operation is equivalent to the keystroke sequence ↰ MTH . The MTH menu is introduced in Chapter 3 (real numbers). Other functions from the MTH menu are presented in Chapters 4 (complex numbers), 8 (lists), 9 (vectors), 10 (matrix creation), 11 (matrix operation), 16 (fast Fourier transforms), 17 (probability applications), and 19 (numbers in different bases).

# CAS menu..

Selecting option *11.CAS menu..* in the APPS menu produces the CAS or SYMBOLIC menu:





This operation is also available by pressing the SYMB key. The CAS or SYMBOLIC menu is introduced in Chapter 5 (algebraic and arithmetic operations). Other functions from the CAS menu are presented in Chapters 4 (complex numbers), 6 (equations solutions), 10 (matrix creation), 11 (matrix operation), 13 (calculus), 14 (multivariate calculus), and 15 (vector analysis).

# Appendix G
# Useful shortcuts

Presented herein are a number of keyboard shortcuts commonly used in the calculator:

- Adjust display contrast: ON (hold) +, or ON (hold) −

- Toggle between RPN and ALG modes: MODE +/- OK

- Set/clear system flag 95 (ALG vs. RPN operating mode)
  MODE FLAGS ▲ ↰ ▲ ↰ ▲ ↰ ▲ CHK

  - In ALG mode,
    CF(-95) selects RPN mode

  - In RPN mode,
    95 +/- ENTER SF selects ALG mode

- A keyboard short cut to toggle between APPROX and EXACT mode is by holding the right-shift key and pressing the ENTER key simultaneously, i.e., ↱ (hold) ENTER.

- Set/clear system flag 105 (EXACT vs. APPROX CAS mode)

  MODE FLAGS ▲ ↰ ▲ ↰ ▲ ▲ ▲ CHK

  - In ALG mode,
    SF(-105) selects APPROX CAS mode
    CF(-105) selects EXACT CAS mode

  - In RPN mode,
    105 +/- ENTER SF selects APPROX CAS mode
    105 +/- ENTER CF selects EXACT CAS mode

- Set/clear system flag 117 (CHOOSE boxes vs. SOFT menus):
  MODE FLAGS ▲ ↰ ▲ ▼ CHK

  - In ALG mode,
    SF(-117) selects SOFT menus
    CF(-117) selects CHOOSE BOXES.

  - In RPN mode,
    117 +/- ENTER SF selects SOFT menus
    117 +/- ENTER CF selects SOFT menus

- Change angular measure:
  - To degrees: ALPHA ALPHA D E G ENTER
  - To radian: ALPHA ALPHA R A D ENTER

- Special characters:
  - Angle symbol (∠): ALPHA ↱ 6
  - Factorial symbol (!): ALPHA ↱ 2
  - Degree symbol (°): ALPHA ↱ (hold) 6

- Lock/unlock alpha keyboard:
  - Lock alpha keyboard (upper case): ALPHA ALPHA
  - Unlock alpha keyboard (upper case): ALPHA
  - Lock alpha keyboard (lower case): ALPHA ALPHA ↰ ALPHA
  - Unlock alpha keyboard (lower case): ↰ ALPHA ALPHA

- Greek letters:

| | | | |
|---|---|---|---|
| Alpha (α): | ALPHA ↱ A | Beta (β): | ALPHA ↱ B |
| DELTA (Δ): | ALPHA ↱ C | Delta (d): | ALPHA ↱ D |
| Epsilon (ε): | ALPHA ↱ E | Rho (ρ): | ALPHA ↱ F |
| Mu (μ): | ALPHA ↱ M | Lambda (λ): | ALPHA ↱ N |
| PI (Π): | ALPHA ↱ P | Sigma (σ): | ALPHA ↱ S |
| Theta (θ): | ALPHA ↱ T | Tau (t): | ALPHA ↱ U |
| Omega (ω): | ALPHA ↱ V | | |

- System-level operation (Hold ON, release it after entering second or third key):

    - ON (hold) F1 F6 : "Cold" restart - all memory erased
    - ON (hold) F2 : Cancels keystroke
    - ON (hold) F3 : "Warm" restart - memory preserved
    - ON (hold) F4 : Starts interactive self-test
    - ON (hold) F5 : Starts continuous self-test
    - ON (hold) SPC : Deep-sleep shutdown - timer off
    - ON (hold) F1 : Performs display screen dump
    - ON (hold) F4 : Cancels next repeating alarm

- Menus not accessible through keyboard: In RPN, enter menu_number, type MENU. In ALG mode, type MENU(menu_number). Menu_number is one of the following:

    - STAT soft menu: 96
    - PLOT soft menu: 81
    - SOLVE soft menu: 74, or use ↱ (hold) 7
    - UTILITY soft menu: 113

- Other menus:

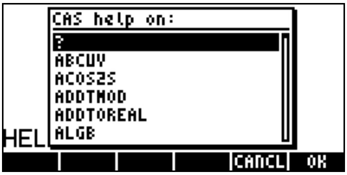
    - MATHS menu: ALPHA ALPHA M A T H S ENTER
    - MAIN menu: ALPHA ALPHA M A I N ENTER

- Other keyboard short cuts:

    - ↱ (hold) 7 : SOLVE menu (menu 74)
    - ↰ (hold) MODE : PRG/MODES menu (Chapter 21)
    - ↰ (hold) ▼ : Starts text editor (Appendix L)
    - ↰ (hold) UPDIR : HOME(), go to HOME directory
    - ↰ (hold) PREV : Recover last active menu
    - ↱ (hold) ▼ : List contents of variables or menu entries
    - ↱ (hold) CHARS : PRG/CHAR menu (Chapter 21)

# Appendix H
# The CAS help facility

The CAS help facility is available through the keystroke sequence TOOL NXT HELP ENTER. The following screen shots show the first menu page in the listing of the CAS help facility.



CAS help on:
?
ABCUV
ACOS2S
ADDTMOD
ADDTOREAL
ALGB

The commands are listed in alphabetical order. Using the vertical arrow keys ▲▼ one can navigate through the help facility list. Some useful hints on navigating through this facility are shown next:

- You can hold down the down arrow key ▼ and watch the screen until the command you're looking for shows up in the screen. At this point, you can release the down arrow key. Most likely the command of interest will not be selected at this point (you may overshoot or undershoot it). However, you can use the vertical keys ▲▼, one stroke at a time, to locate the command you want, and then press OK.
- If, while holding down the down arrow key ▼ you overshoot the command of interest, you can hold down the up arrow key ▲ to move back towards that command. Refine the selection with the vertical keys ▲▼, one stroke at a time.
- You can type the first letter of the command of interest, and then use the down arrow key ▼ to locate that particular command. For example, if you're looking for the command DERIV. After activating the help facility (TOOL NXT HELP ENTER), type ALPHA D. This will select the first of the commands that start with D, i.e., DEGREE. To find DERIV, press ▼, twice. To activate the command, press OK.

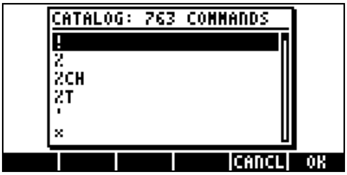
- You can type two or more letters of the command of interest, by locking the alphabetic keyboard. This will take you to the command of interest, or to its neighborhood. Afterwards, you need to unlock the alpha keyboard, and use the vertical arrow keys ▲ ▼ to locate the command, if needed. Press OK to locate the to activate the command. For example, to locate the command PROPFRAC, you can use, one of the following keystroke sequences:

  TOOL NXT HELP ENTER ALPHA ALPHA P R ALPHA ▼ ▼ OK
  TOOL NXT HELP ENTER ALPHA ALPHA P R O ALPHA ▼ OK
  TOOL NXT HELP ENTER ALPHA ALPHA P R O P ALPHA OK

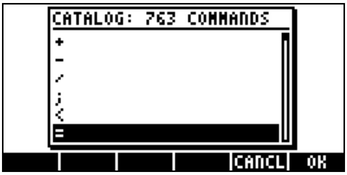
See Appendix C for more information on the CAS (Computer Algebraic System). Appendix C includes other examples of application of the CAS help facility.

# Appendix I
# Command catalog list

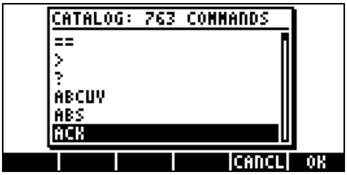
This is a list of all commands in the command catalog (→ CAT ). Those commands that belong to the CAS (Computer Algebraic System) are listed also in Appendix H. CAS help facility entries are available for a given command if the soft menu key HELP shows up when you highlight that particular command. Press this soft menu key to get the CAS help facility entry for the command. The first few screens of the catalog are shown below:



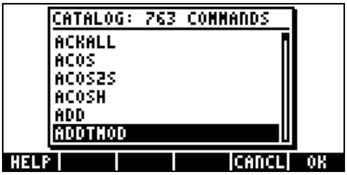
```
CATALOG: 763 COMMANDS
!
%
%CH
%T
'
×
          CANCL  OK
```



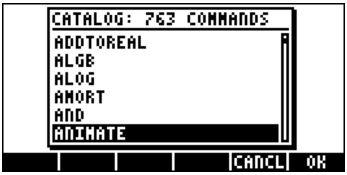
```
CATALOG: 763 COMMANDS
+
-
/
;
<
=
          CANCL  OK
```



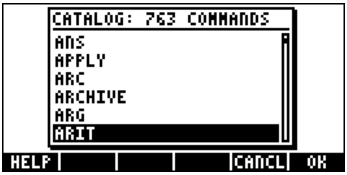
```
CATALOG: 763 COMMANDS
==
>
?
ABCUV
ABS
ACK
          CANCL  OK
```



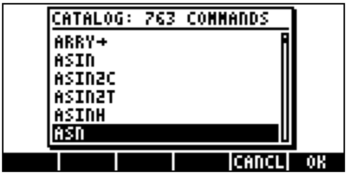
```
CATALOG: 763 COMMANDS
ACKALL
ACOS
ACOS2S
ACOSH
ADD
ADDTMOD
HELP      CANCL  OK
```



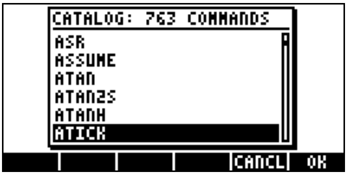
```
CATALOG: 763 COMMANDS
ADDTOREAL
ALGB
ALOG
AMORT
AND
ANIMATE
          CANCL  OK
```



```
CATALOG: 763 COMMANDS
ANS
APPLY
ARC
ARCHIVE
ARG
ARIT
HELP      CANCL  OK
```



```
CATALOG: 763 COMMANDS
ARRY→
ASIN
ASIN2C
ASIN2T
ASINH
ASN
          CANCL  OK
```

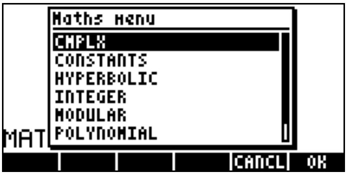


```
CATALOG: 763 COMMANDS
ASR
ASSUME
ATAN
ATAN2S
ATANH
ATICK
          CANCL  OK
```

User-installed library commands would also appear on the command catalog list, using italic font. If the library includes a help item, then the soft menu key HELP shows up when you highlight those user-created commands.

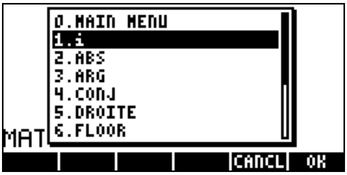
# Appendix J
# The MATHS menu

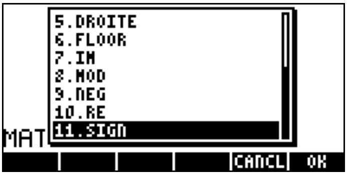
The MATHS menu, accessible through the command MATHS (available in the catalog CAT ), contains the following sub-menus:



## The CMPLX sub-menu

The CMPLX sub-menu contains functions pertinent to operations with complex numbers:

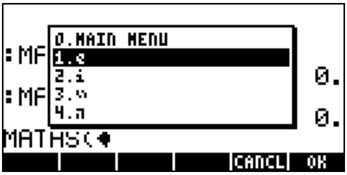




These functions are described in Chapter 4.

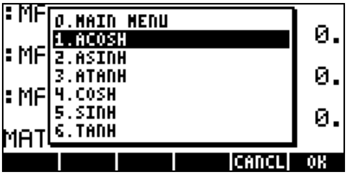
## The CONSTANTS sub-menu

The CONSTANTS sub-menu provides access to the calculator mathematical constants. These are described in Chapter 3:



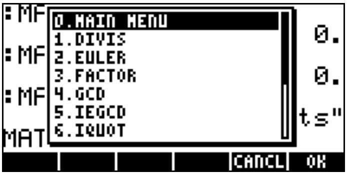
## The HYPERBOLIC sub-menu

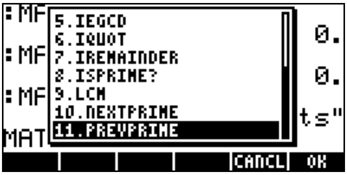
The HYPERBOLIC sub-menu contains the hyperbolic functions and their inverses. These functions are described in Chapter 3.



## The INTEGER sub-menu

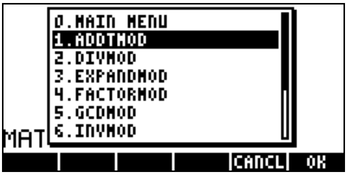
The INTEGER sub-menu provides functions for manipulating integer numbers and some polynomials. These functions are presented in Chapter 5:

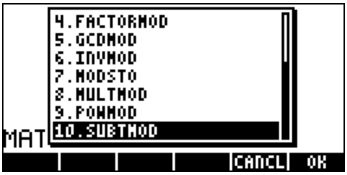




## The MODULAR sub-menu

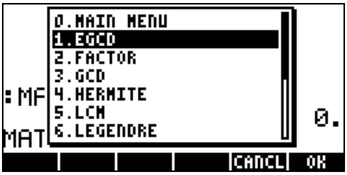
The MODULAR sub-menu provides functions for modular arithmetic with numbers and polynomials. These functions are presented in Chapter 5:

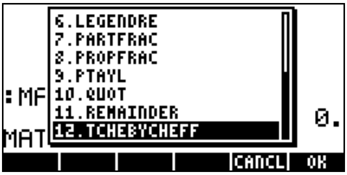




## The POLYNOMIAL sub-menu

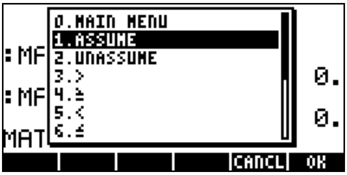
The POLYNOMIAL sub-menu includes functions for generating and manipulating polynomials. These functions are presented in Chapter 5:

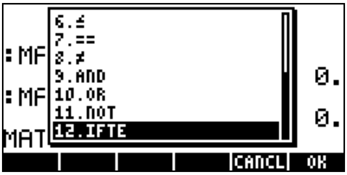




## The TESTS sub-menu

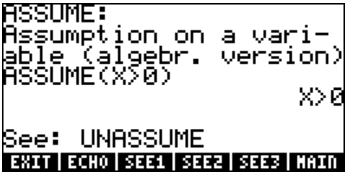
The TESTS sub-menu includes relational operators (e.g., ==, <, etc.), logical operators (e.g., AND, OR, etc.), the IFTE function, and the ASSUME and UNASSUME commands.



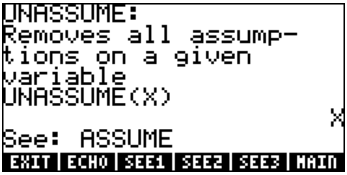


Relational and logical operators are presented in Chapter 21 in the context of programming the calculator in User RPL language. The IFTE function is introduced in Chapter 3. Functions ASSUME and UNASSUME are presented next, using their CAS help facility entries (see Appendix C).

### ASSUME

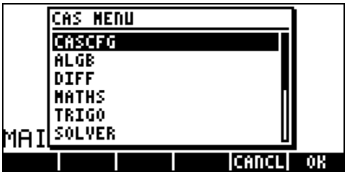

```
ASSUME:
Assumption on a vari-
able (algebr. version)
ASSUME(X>0)
                    X>0

See: UNASSUME
```

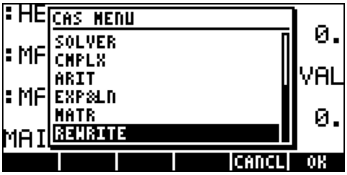
### UNASSUME


```
UNASSUME:
Removes all assump-
tions on a given
variable
UNASSUME(X)
                      X
See: ASSUME
```

# Appendix K
# The MAIN menu

The MAIN menu is available in the command catalog.  This menu include the following sub-menus:



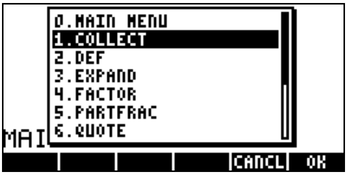


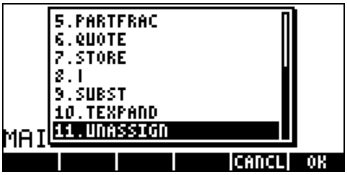
## The CASCFG command

This is the first entry in the MAIN menu.  This command configures the CAS. For CAS configuration information see Appendix C.

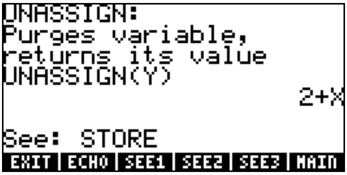
### The ALGB sub-menu

The ALGB sub-menu includes the following commands:





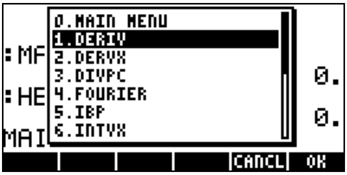
These functions, except for 0.MAIN MENU and 11.UNASSIGN are available in the ALG keyboard menu (↱ ALG ).  Detailed explanation of these functions can be found in Chapter 5.   Function UNASSIGN is described in the following entry from the CAS menu:

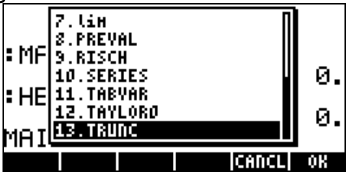


```
UNASSIGN:
Purges variable,
returns its value
UNASSIGN(Y)
                 2+X

See: STORE
```

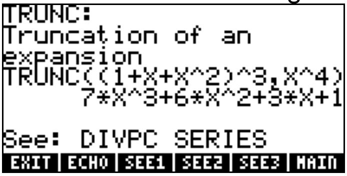
## The DIFF sub-menu

The DIFF sub-menu contains the following functions:





These functions are also available through the CALC/DIFF sub-menu (start with ↰ CALC ). These functions are described in Chapters 13, 14, and 15, except for function TRUNC, which is described next using its CAS help facility entry:



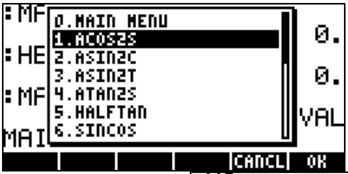
```
TRUNC:
Truncation of an
expansion
TRUNC((1+X+X^2)^3,X^4)
     7*X^3+6*X^2+3*X+1

See: DIVPC SERIES
```

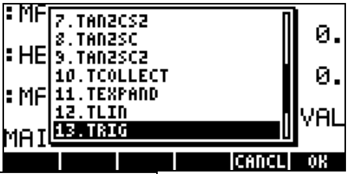
# The MATHS sub-menu

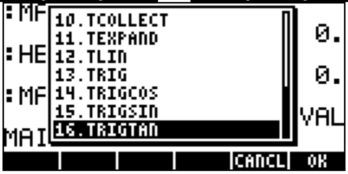
The MATHS menu is described in detail in Appendix J.

## The TRIGO sub-menu

The TRIGO menu contains the following functions:



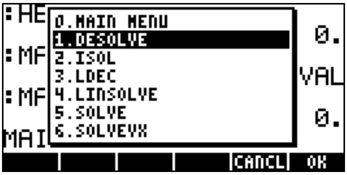




These functions are also available in the TRIG menu (↱ TRIG ). Description of these functions is included in Chapter 5.

## The SOLVER sub-menu

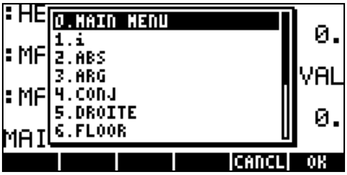
The SOLVER menu includes the following functions:

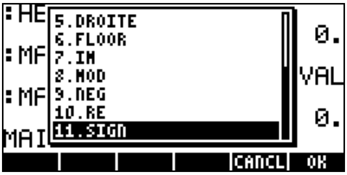


These functions are available in the CALC/SOLVE menu (start with ↰ CALC ). The functions are described in Chapters 6, 11, and 16.

## The CMPLX sub-menu

The CMPLX menu includes the following functions:

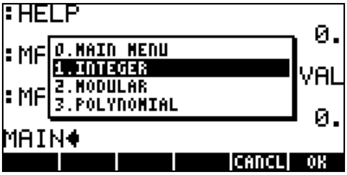




The CMPLX menu is also available in the keyboard (↱ CMPLX ). Some of the functions in CMPLX are also available in the MTH/COMPLEX menu (start with ↰ MTH ). Complex number functions are presented in Chapter 4.

# The ARIT sub-menu

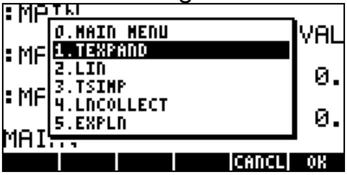
The ARIT menu includes the following sub-menus:



The sub-menus INTEGER, MODULAR, and POLYNOMIAL are presented in detail in Appendix J.

## The EXP&LN sub-menu

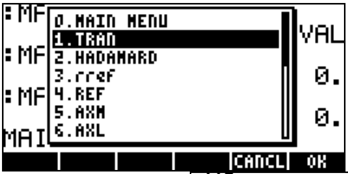
The EXP&LN menu contains the following functions:

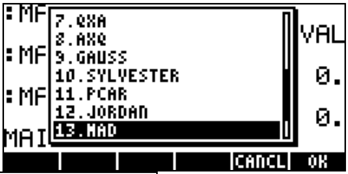


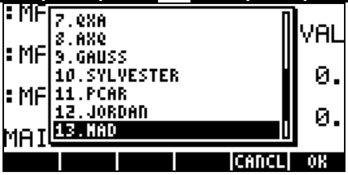
This menu is also accessible through the keyboard by using ← EXP&LN . The functions in this menu are presented in Chapter 5.

## The MATR sub-menu

The MATR menu contains the following functions:



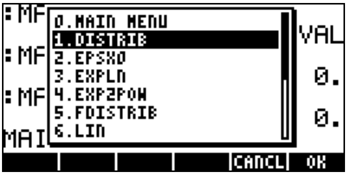


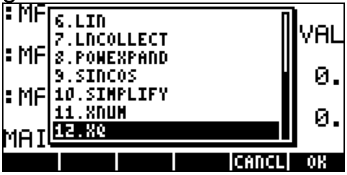


These functions are also available through the MATRICES menu in the keyboard (← MATRICES ). The functions are described in Chapters 10 and 11.

### The REWRITE sub-menu

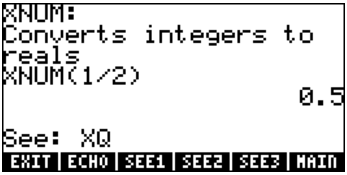
The REWRITE menu contains the following functions:





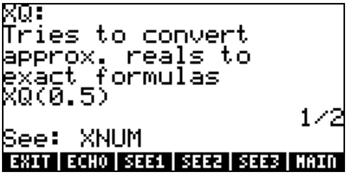
These functions are available through the CONVERT/REWRITE menu (start with ← CONVERT ). The functions are presented in Chapter 5, except for functions XNUM and XQ, which are described next using the corresponding entries in the CAS help facility (TOOL NXT HELP ):

### XNUM



```
XNUM:
Converts integers to
reals
XNUM(1/2)
                   0.5

See: XQ
```

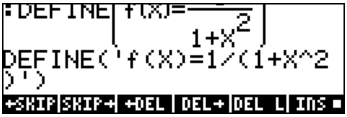
### XQ



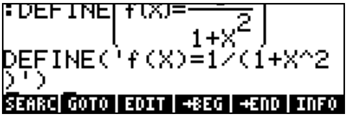
```
XQ:
Tries to convert
approx. reals to
exact formulas
XQ(0.5)
                   1/2
See: XNUM
```

# Appendix L
# Line editor commands

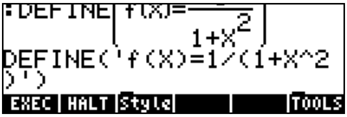
When you trigger the line editor by using ↰ ▼ in the RPN stack or in ALG mode, the following soft menu functions are provided (press NXT to see the remaining functions):



```
:DEFINE f(X)=
              1+X^2
DEFINE('f(X)=1/(1+X^2
)')
```



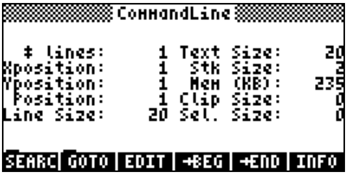
```
:DEFINE f(X)=
              1+X^2
DEFINE('f(X)=1/(1+X^2
)')
```



```
:DEFINE f(X)=
              1+X^2
DEFINE('f(X)=1/(1+X^2
)')
```

The functions are briefly described as follows:

←SKIP: Skips characters to beginning of word.
SKIP→: Skips characters to end of word.
←DEL: Delete characters to beginning of word.
DEL→: Delete characters to end of word.
DEL L: Delete characters in line.
INS: When selected inserts characters at cursor location. If not selected, the cursor replaces characters (overwrites) instead of inserting characters.
EDIT: Edits selection.
→BEG: Move to beginning of word.
→END: Mark end of selection.
INFO: Provides information on Command Line editor, e.g.,



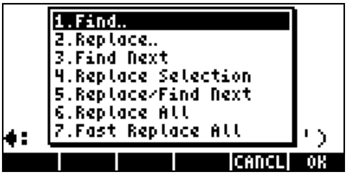
```
             CommandLine

  # lines:      1 Text Size:   20
Xposition:      1  Stk Size:    2
Yposition:      1  Mem (KB):  235
 Position:      1 Clip Size:    0
Line Size:     20 Sel. Size:    0
```

The items show in this screen are self-explanatory. For example, *X* and Y *positions* mean the position on a line (X) and the line number (Y). *Stk Size* means the number of objects in the ALG mode history or in the RPN stack. *Mem(KB)* means the amount of free memory. *Clip Size* is the number of characters in the clipboard. *Sel Size* is the number of characters in the current selection.

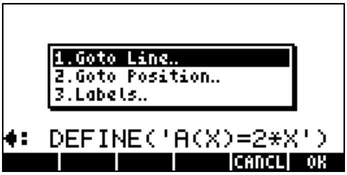
EXEC: Execute command selected.
HALT: Stop command execution.

The line editor also provide the following sub-menus:

SEARCH: Search characters or words in the command line. It includes the following functions:

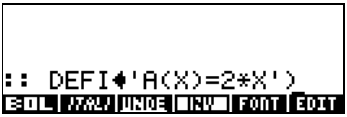


GOTO: Move to a desired location in the command line. It includes the following functions:



DEFINE('A(X)=2*X')

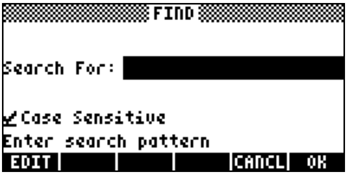
Style: Text styles that can be used in the command line:


:: DEFI◆'A(X)=2*X')

## The SEARCH sub-menu

The functions of the SEARCH sub-menu are:

*Find* : Use this function to find a string in the command line.  The input form provided with this command is shown next:



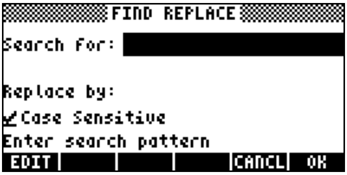
FIND

Search For:

☑ Case Sensitive

Enter search pattern

*Replace*: Use this command to find and replace a string. The input form provided for this command is:



FIND REPLACE

Search For:

Replace by:

☑ Case Sensitive

Enter search pattern

*Find next..*: Finds the next search pattern as defined in Find

*Replace Selection*: Replace selection with replacement pattern defined with Replace command.

*Replace/Find Next*: Replace a pattern and search for another occurrence. The pattern is defined in Replace.

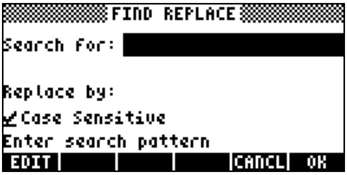
*Replace All*: Replace all occurrence of a certain pattern. This command asks for confirmation from the user before replacing pattern.

*Fast Replace All*: Replace all occurrences of a certain pattern without checking with the user.

## The GOTO sub-menu

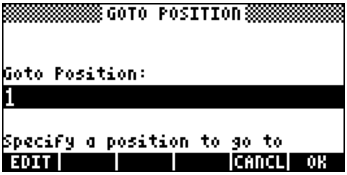
The functions in the GOTO sub-menu are the following:

*Goto Line:* to move to a specified line. The input form provided with this command is:



```
FIND REPLACE
Search for:

Replace by:
✓Case Sensitive
Enter search pattern
```

*Goto Position*: move to a specified position in the command line. The input form provided for this command is:



```
GOTO POSITION

Goto Position:
1

Specify a position to go to
```

*Labels*: move to a specified label in the command line.

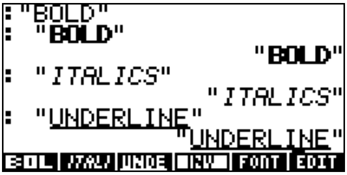
## The Style sub-menu

The Style sub-menu includes the following styles:

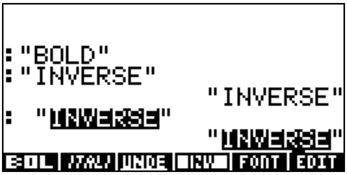
**BOL**: Bold
*ITALI*: Italics
UNDE: Underline
INV : Inverse

The command FONT allow the user to select the font for the command editor.

Examples of the different styles are shown below:



```
:"BOLD"
: "BOLD"
                  "BOLD"
: "ITALICS"
             "ITALICS"
: "UNDERLINE"
           "UNDERLINE"
```



```
:"BOLD"
:"INVERSE"
             "INVERSE"
: "INVERSE"
             "INVERSE"
```

# Appendix M
# Index

# Limited Warranty

hp 49g+ graphing calculator; Warranty period: 12 months

1. HP warrants to you, the end-user customer, that HP hardware, accessories and supplies will be free from defects in materials and workmanship after the date of purchase, for the period specified above. If HP receives notice of such defects during the warranty period, HP will, at its option, either repair or replace products which prove to be defective. Replacement products may be either new or like-new.
2. HP warrants to you that HP software will not fail to execute its programming instructions after the date of purchase, for the period specified above, due to defects in material and workmanship when properly installed and used. If HP receives notice of such defects during the warranty period, HP will replace software media which does not execute its programming instructions due to such defects.
3. HP does not warrant that the operation of HP products will be uninterrupted or error free. If HP is unable, within a reasonable time, to repair or replace any product to a condition as warranted, you will be entitled to a refund of the purchase price upon prompt return of the product with proof of purchase.
4. HP products may contain remanufactured parts equivalent to new in performance or may have been subject to incidental use.
5. Warranty does not apply to defects resulting from (a) improper or inadequate maintenance or calibration, (b) software, interfacing, parts or supplies not supplied by HP, (c) unauthorized modification or misuse, (d) operation outside of the published environmental specifications for the product, or (e) improper site preparation or maintenance.
6. HP MAKES NO OTHER EXPRESS WARRANTY OR CONDITION WHETHER WRITTEN OR ORAL. TO THE EXTENT ALLOWED BY LOCAL LAW, ANY IMPLIED WARRANTY OR CONDITION OF MERCHANTABILITY, SATISFACTORY QUALITY, OR FITNESS FOR A PARTICULAR PURPOSE IS LIMITED TO THE DURATION OF THE EXPRESS WARRANTY SET FORTH ABOVE. Some countries, states or provinces do not allow limitations on the duration of an implied warranty, so the above limitation or exclusion might not apply to you. This warranty gives you specific legal rights and you might also have other rights that vary from country to country, state to state, or province to province.

7. TO THE EXTENT ALLOWED BY LOCAL LAW, THE REMEDIES IN THIS WARRANTY STATEMENT ARE YOUR SOLE AND EXCLUSIVE REMEDIES. EXCEPT AS INDICATED ABOVE, IN NO EVENT WILL HP OR ITS SUPPLIERS BE LIABLE FOR LOSS OF DATA OR FOR DIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL (INCLUDING LOST PROFIT OR DATA), OR OTHER DAMAGE, WHETHER BASED IN CONTRACT, TORT, OR OTHERWISE. Some countries, States or provinces do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.
8. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. HP shall not be liable for technical or editorial errors or omissions contained herein.

**FOR CONSUMER TRANSACTIONS IN AUSTRALIA AND NEW ZEALAND: THE WARRANTY TERMS CONTAINED IN THIS STATEMENT, EXCEPT TO THE EXTENT LAWFULLY PERMITTED, DO NOT EXCLUDE, RESTRICT OR MODIFY AND ARE IN ADDITION TO THE MANDATORY STATUTORY RIGHTS APPLICABLE TO THE SALE OF THIS PRODUCT TO YOU.**

# Service

**Europe**

| Country : | Telephone numbers |
|---|---|
| Austria | +43-1-3602771203 |
| Belgium | +32-2-7126219 |
| Denmark | +45-8-2332844 |
| Eastern Europe countries | +420-5-41422523 |
| Finland | +35-89640009 |
| France | +33-1-49939006 |
| Germany | +49-69-95307103 |
| Greece | +420-5-41422523 |
| Holland | +31-2-06545301 |
| Italy | +39-0422-303069 |
| Norway | +47-63849309 |
| Portugal | +351-213-180020 |
| Spain | +34-917-820111 |

| | |
|---|---|
| Sweden | +46-851992065 |
| Switzerland | +41-1-4395358 (German) |
| | +41-22-8278780 (French) |
| | +39-0422-303069 (Italian) |
| Turkey | +420-5-41422523 |
| UK | +44-207-4580161 |
| Czech Republic | +420-5-41422523 |
| South Africa | +27-11-541 9573 |
| Luxembourg | +32-2-7126219 |
| Other European countries | +420-5-41422523 |

**Asia Pacific**

| Country : | Telephone numbers |
|---|---|
| Australia | +61-3-9841-5211 |
| Singapore | +61-3-9841-5211 |

**L.America**

| Country : | Telephone numbers |
|---|---|
| Argentina | 0-810-555-5520 |
| | Sao Paulo 3747-7799; ROTC |
| Brazil | 0-800-157751 |
| | Mx City 5258-9922; ROTC |
| Mexico | 01-800-472-6684 |
| Venezuela | 0800-4746-8368 |
| Chile | 800-360999 |
| Columbia | 9-800-114726 |
| Peru | 0-800-10111 |
| Central America & Caribbean | 1-800-711-2884 |
| Guatemala | 1-800-999-5105 |
| Puerto Rico | 1-877-232-0589 |
| Costa Rica | 0-800-011-0524 |

**N.America**

| Country : | Telephone numbers |
|---|---|
| U.S. | 1800-HP INVENT |
| Canada | (905) 206-4663 or 800- HP INVENT |

ROTC = Rest of the country

## Regulatory information

This section contains information that shows how the hp 49g+ graphing calculator complies with regulations in certain regions. Any modifications to the calculator not expressly approved by Hewlett-Packard could void the authority to operate the 49g+ in these regions.

### USA

This calculator generates, uses, and can radiate radio frequency energy and may interfere with radio and television reception. The calculator complies with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation.
However, there is no guarantee that interference will not occur in a particular installation. In the unlikely event that there is interference to radio or television reception(which can be determined by turning the calculator off and on), the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Relocate the calculator, with respect to the receiver.

#### Connections to Peripheral Devices

To maintain compliance with FCC rules and regulations, use only the cable accessories provided.

### Canada

This Class B digital apparatus complies with Canadian ICES-003. Cet appareil numerique de la classe B est conforme a la norme NMB-003 du Canada.

### Japan

　この装置は、情報処理装置等電波障害自主規制協議会(VCCI)の基準に基づく第二情報技術装置です。この装置は、家庭環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。

取扱説明書に従って正しい取り扱いをしてください。