

# 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,  $n \times m$  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  $\mathbf{A} = [a_{ij}]_{n \times m}$ . The full matrix is shown next:

$$\mathbf{A} = [a_{ij}]_{n \times m} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}.$$

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  $3 \times 3$  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

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}.$$

# Entering matrices in the stack

In this section we present two different methods to enter matrices in the calculator stack: (1) using the Matrix Writer, and (2) typing the matrix directly into the stack.

## Using the Matrix Writer

As with the case of vectors, discussed in Chapter 9, matrices can be entered into the stack by using the Matrix Writer. For example, to enter the matrix:

$$\begin{bmatrix} -2.5 & 4.2 & 2.0 \\ 0.3 & 1.9 & 2.8 \\ 2 & -0.1 & 0.5 \end{bmatrix},$$

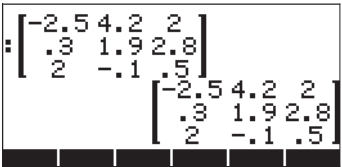
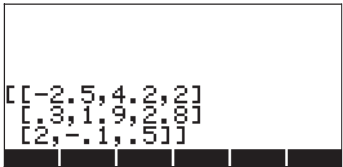
first, start the matrix writer by using  $\leftarrow$  *MTRW*. Make sure that the option  $\leftarrow \rightarrow$  is selected. Then use the following keystrokes:


$\left[ 2 \right] \left[ . \right] \left[ 5 \right] \left[ +/- \right] \left[ ENTER \right] \left[ 4 \right] \left[ . \right] \left[ 2 \right] \left[ ENTER \right] \left[ 2 \right] \left[ ENTER \right] \left[ \nabla \right] \left[ \leftarrow \right] \left[ \leftarrow \right] \left[ \leftarrow \right] \left[ . \right] \left[ 3 \right] \left[ ENTER \right] \left[ / \right] \left[ . \right] \left[ 9 \right] \left[ ENTER \right] \left[ 2 \right] \left[ . \right] \left[ 8 \right] \left[ ENTER \right] \left[ 2 \right] \left[ ENTER \right] \left[ . \right] \left[ / \right] \left[ +/- \right] \left[ ENTER \right] \left[ . \right] \left[ 5 \right] \left[ ENTER \right]$

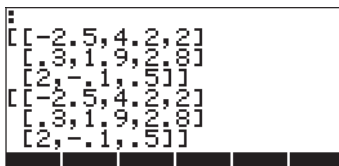
At this point, the Matrix Writer screen may look like this:



Press  $\left[ ENTER \right]$  once more to place the matrix on the stack. The ALG mode stack is shown next, before and after pressing  $\left[ ENTER \right]$ , once more:



If you have selected the textbook display option (using **MODE**  and checking off  $\checkmark$ Textbook), the matrix will look like the one shown above. Otherwise, the display will show:



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:

$\leftarrow$   $\left[ \right]$   $\leftarrow$   $\left[ \right]$   $\leftarrow$  2  $\leftarrow$  .  $\leftarrow$  5  $\leftarrow$  +/-  $\leftarrow$  ,  $\leftarrow$  4  $\leftarrow$  .  $\leftarrow$  2  $\leftarrow$  ,  $\leftarrow$  2  $\leftarrow$   $\rightarrow$   
 $\leftarrow$  ,  
 $\leftarrow$   $\left[ \right]$   $\leftarrow$  .  $\leftarrow$  3  $\leftarrow$  ,  $\leftarrow$  1  $\leftarrow$  .  $\leftarrow$  9  $\leftarrow$  ,  $\leftarrow$  2  $\leftarrow$  .  $\leftarrow$  8  $\leftarrow$   $\rightarrow$   
 $\leftarrow$  ,  
 $\leftarrow$   $\left[ \right]$   $\leftarrow$  2  $\leftarrow$  ,  $\leftarrow$  .  $\leftarrow$  1  $\leftarrow$  +/-  $\leftarrow$  ,  $\leftarrow$  .  $\leftarrow$  5

Thus, to enter a matrix directly into the stack open a set of brackets ( $\leftarrow$   $\left[ \right]$  ) and enclose each row of the matrix with an additional set of brackets ( $\leftarrow$   $\left[ \right]$  ). Commas ( $\leftarrow$  ,  $\leftarrow$  . ) 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,  $\left[ \left[ 1 \ 2 \ 3 \right] \left[ 4 \ 5 \ 6 \right] \left[ 7 \ 8 \ 9 \right] \right]$ , type  $\left[ 1 \ 2 \ 3 \right] 4 \ 5 \ 6 \ 7 \ 8 \ 9$ .)

For future exercises, let's save this matrix under the name A. In ALG mode use  $\leftarrow$   $\left[ \right]$   $\leftarrow$  ALPHA  $\leftarrow$  A . In RPN mode, use  $\leftarrow$  '  $\leftarrow$  ALPHA  $\leftarrow$  A  $\leftarrow$   $\leftarrow$   $\leftarrow$  .

## 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 ( $\leftarrow$   $\leftarrow$  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,  $\rightarrow$ DIAG, and DIAG $\rightarrow$ . 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:  $\leftarrow$  MTH  $\rightarrow$   $\rightarrow$   $\rightarrow$

The functions available will be shown as soft-menu key labels as follows (press  $\rightarrow$  NEXT to move to the next set of functions):



With system flag 117 set to SOFT menus, the functions of the CREATE menu, triggered by  $\leftarrow$  MATRICES  $\rightarrow$   $\rightarrow$   $\rightarrow$ , 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))
: A(2,3)
2.8
2.8
GET | GETI | PUT | PUTI | SUB | REPL
```

Notice that we achieve the same result by simply typing  $A(2,3)$  and pressing  $\text{ENTER}$ . In RPN mode, this exercise is performed by entering  $\text{RPN} \text{ ENTER } 3 \text{ ENTER GET}$ , or by using  $A(2,3) \text{ ENTER}$ .

Suppose that we want to place the value ' $\pi$ ' into element  $a_{31}$  of the matrix. We can use function PUT for that purpose, e.g.,

```
:PUT(A,(3 1),pi)
-2.5 4.2 2
.3 1.9 2.8
pi -.1 .5
GET | GETI | PUT | PUTI | SUB | REPL
```

In RPN mode you can use:  $\text{VAR} \text{ RPN } (3,1) \text{ ENTER } \leftarrow \pi \text{ PUT}$ . Alternatively, in RPN mode you can use:  $\leftarrow \pi \text{ 'A(2,3)' ENTER STO}$ . To see the contents of variable A after this operation, use  $\text{RPN}$ .

## 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:  $\text{RPN} \{2,2\} \text{ ENTER GETI}$ . Screen shots showing the RPN stack before and after the application of function GETI are shown below:

```

3: [-2.5 4.2 2]
2: [.3 1.9 π]
1: [ 2 -1.5]
   {2 2}
A

```

```

3: [-2.5 4.2 2]
2: [.3 1.9 π]
1: [ 2 -1.5]
   {2 3}
1.9
GET GETI PUT PUTI SUB REPL

```

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:  $\leftarrow \leftarrow \{3\ 1\} \text{ (ENTER) } 2 \text{ (ENTER)}$  PUTI. The screen shots below show the RPN stack before and after the application of function PUTI:

```

3: [-2.5 4.2 2]
2: [.3 1.9 π]
1: [ 2 -1.5]
   {3 1}
2

```

```

3: [-2.5 4.2 2]
2: [.3 1.9 π]
1: [ 2 -1.5]
   {3 2}
2
GET GETI PUT PUTI SUB REPL

```

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.)
CON IDN TRN RDN RANH SIZE

```

In RPN mode, these exercises are performed by using  $\leftarrow \leftarrow \text{SIZE}$ , and  $\leftarrow \leftarrow [1,2], [3,4] \text{ (ENTER) SIZE}$ .

## 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]

CON IDN TRN RDM RANM SIZE

```

```


: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-i:2]
      [4.2-4.2-i:2 1.9-1.9-i:2 -1--1-i:2]
      [2-2-i:2 π-π-i:2 .5-.5-i:2]

CON IDN TRN RDM RANM SIZE

```

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 TRN.

**Note:** The calculator also includes Function TRAN in the MATRICES/OPERATIONS sub-menu:

```

RAD CHOM
MATRICES MENU
1.CREATE..
2.OPERATIONS..
3.FACTORIZATION..
4.QUADRATIC FORM..
5.LINEAR SYSTEMS..
6.LINEAR APPL..
7.EIGENVECTORS..
8.VECTOR..
      [CANCEL] [OK]

```

```

RAD CHOM
MATRIX OPERATIONS MENU
11.RANM
12.RSD
13.SIZE
14.SRANM
15.SRAD
16.TRACE
17.TRAN
18.MATRICES..
      [HELP] [CANCEL] [OK]

```

For example, in ALG mode:

```

:TRAN(A)
      [-2.5 .3 2]
      [4.2 1.9 -1]
      [2 π .5]

A

```

## 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
+SKIP SKIP+ +DEL DEL+ DEL L IN$

```

In RPN mode this is accomplished by using  $\{4, 3\}$  ENTER / . 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 4x4 identity matrix in ALG mode use:

```

: IDN(4)
  1 0 0 0
  0 1 0 0
  0 0 1 0
  0 0 0 1
CON | IDN | TRN | RDM | RDM | SIZE

```

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
CON | IDN | TRN | RDM | RDM | SIZE

```

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 A 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]
CON | IDN | TRN | RDM | RANM | SIZE
```

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:


```
:RDM([1 2 3 4 5 6],(2 3))
      [1 2 3]
      [4 5 6]
:RDM(ANS(1),(3 2))
      [1 2]
      [3 4]
      [5 6]
CON | IDN | TRN | RDM | RANM | SIZE
```

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:

```
:RDM(ANS(1),(3 2))
      [1 2]
      [3 4]
      [5 6]
:RDM(ANS(1),(6))
      [1 2 3 4 5 6]
CON | IDN | TRN | RDM | RANM | SIZE
```


If using RPN mode, we assume that the matrix is in the stack and use {6}  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  $2 \times 3$  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]
CON | IDN | TRN | RDM | RANM | SIZE
```

In RPN mode, use {2,3}  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

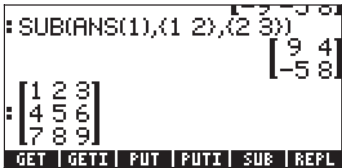
Function SUB extracts a sub-matrix from an existing matrix, provided you indicate the initial and final position of the sub-matrix. For example, if we want to extract elements  $a_{12}$ ,  $a_{13}$ ,  $a_{22}$ , and  $a_{23}$  from the last result, as a  $2 \times 2$  sub-matrix, in ALG mode, use:

```
:RANM((2 3))      [ 2  5  0]
                  [-4  9  4]
                  [-9 -5  8]
:SUB(ANS(1),(1 2),(2 3))
                  [ 9  4]
                  [-5  8]
GET | GETI | PUT | PUTI | SUB | REPL
```

In RPN mode, assuming that the original 2x3 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 2x2 matrix, into the 3x3 matrix currently located in **ANS(1)**, starting at position (2,2):

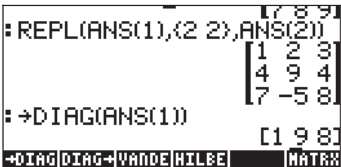


If working in the RPN mode, assuming that the 2x2 matrix was originally in the stack, we proceed as follows:

[[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 x 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:



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:



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:



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 from 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<sub>1</sub>, x<sub>2</sub>,... x<sub>n</sub>}, 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
+DIAGDIAG+VANDERMONDE+HILBERT+MATRIX

```

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 **COLS**) or rows of the matrix (program **ROWS**). 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 `mat` allows you to put together a  $p \times n$  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:

1 > << >>  
 2 > < PRG STACK SWAP  
 3 > > SPC ALPHA < (N  
 4 > << >>  
 5 > < PRG STACK SWAP  
 6 > < PRG BRANCH FOR FOR  
 7 ALPHA < < /  
 8 > < PRG TYPE 031 ->  
 9 > > STACK  
 10 > < PRG BRANCH IF IF  
 11 ALPHA < < / SPC  
 12 ALPHA < < (N  
 13 > < PRG TEST <  
 14 > < PRG BRANCH IF THEN  
 15 ALPHA < < / SPC / +  
 16 > < PRG STACK NEXT ROLL  
 17 > < PRG BRANCH IF END  
 18 > < PRG BRANCH FOR NEXT  
 19 > < PRG BRANCH IF IF  
 20 ALPHA < < (N SPC /  
 21 > < PRG TEST >  
 22 > < PRG BRANCH IF THEN  
 23 > / SPC  
 24 ALPHA < < (N SPC / -  
 25 > < PRG BRANCH FOR FOR  
 26 ALPHA < < / SPC  
 27 ALPHA < < / SPC / +  
 28 > < PRG STACK NEXT ROLL  
 29 > < PRG BRANCH FOR NEXT  
 30 > < PRG BRANCH IF END

Produces:

```

«
DUP
→ n
<<
1 SWAP
FOR
i
OBJ→
→ARRAY
IF
i
n
<
THEN
i 1 +
ROLL
END
NEXT
IF
n 1
>
THEN
1
n 1 -
FOR
i
i 1 +
ROLL
NEXT
END

```


n  
COL→  
Program is displayed in level 1

1 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   . The program listing is the following:




```
« DUP → n « 1 SWAP FOR j OBJ → ARRAY 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 . As an example, try the following exercise:

(1,2,3,4) (ENTER) (1,4,9,16) (ENTER) (1,8,27,64) (ENTER) 3 (ENTER) 

The following screen shots show the RPN stack before and after running program **4310**:

1:						[	1	1	1]
							2	4	8]
							3	9	27]
							4	16	64]
CRC	0								

To use the program in ALG mode, press  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:

```

:CRMCM(1 2 3 4),(1 4 9 16),M
      1 1 1
      2 4 8
      3 9 27
      4 16 64
HYP ACOS ASIN ASINH ATANH HALT

```



### 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:

→

EDIT

↓

→

↓

↑

←

←

←

←

←

←

ALPHA

ALPHA

(R)

(O)

(W)

ALPHA

ENTER

List program CRMC in stack

Move to end of program

Delete COL

Type in ROW, enter program

To store the program use: 

⏏

ALPHA

ALPHA

C

(R)

(M)

(R)

ALPHA

STOP

(1,2,3,4)

ENTER

(1,4,9,16)

ENTER

(1,8,27,64)

ENTER

3

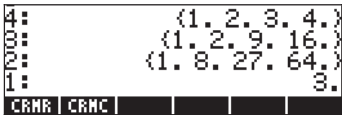
ENTER

EDIT

The following screen shots show the RPN stack before and after running program 

EDIT

:



These programs can be useful for statistical applications, specifically to create the statistical matrix  $\Sigma$  DAT. 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:



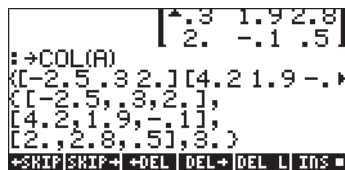
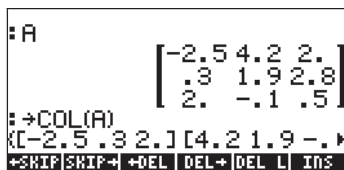
When system flag 117 is set to SOFT menus, the COL menu is accessible through  $\leftarrow$  MTH  $\leftarrow$  MATRX  $\leftarrow$  COL, or through  $\leftarrow$  MATRICES  $\leftarrow$  CREATE  $\leftarrow$  COL. Both approaches will show the same set of functions:



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  $\nabla$ ).



In RPN mode, you need to list the matrix in the stack, and the activate function →COL, i.e.,  $\leftarrow$  →COL. The figure below shows the RPN stack before and after the application of function →COL.

7:					
6:					
5:					
4:					
3:					
2:					
1:					
0:					
CRNR	CRNC	A			

7:					
6:					
5:					
4:					
3:					
2:					
1:					
0:					
CRNR	CRNC	A			

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)

7:					
6:					
5:					
4:					
3:					
2:					
1:					
0:					
CRNR	CRNC	A			

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:					
3:					
2:					
1:					
0:					
CRNR	CRNC	A			

4:					
3:					
2:					
1:					
0:					
CRNR	CRNC	A			

## 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. -3.1,2.])
[-2.5 -1. 4.2 2.]
[.3 -2. 1.9 2.8]
[2. -3. -.1 .5]
CRNR | CRMC | A |

```

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.]
2: [.3 1.9 2.8]
1: [2. -.1 .5]
: [-1. -2. -3.1]
CRNR | CRMC | A |

```

```

4:
3: [-2.5 -1. 4.2 2.]
2: [.3 -2. 1.9 2.8]
1: [2. -3. -.1 .5]
CRNR | CRMC | A |

```

## 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] [2. 2.8 .5] }
{ [.3 1.9] }
{ [2. -.1] }
CRNR | CRMC | A |

```

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.]
1: [.3 1.9 2.8]
: [2. -.1 .5]
CRNR | CRMC | A |

```

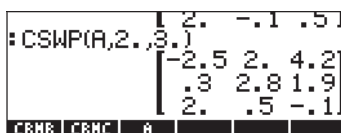
```

2: [-2.5 4.2]
1: [.3 1.9]
: [2. 2.8 .5]
CRNR | CRMC | A |

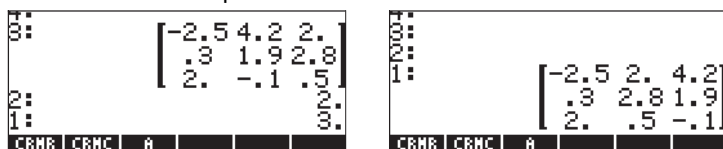
```

## 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.



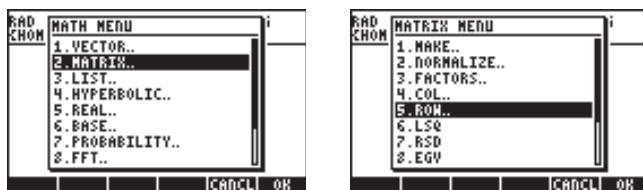
In RPN mode, function CSWP lets you swap the columns of a matrix listed in stack level 3, whose indices are listed in stack levels 1 and 2. For example, the following figure shows the RPN stack before and after applying function CSWP to matrix A in order to swap columns 2 and 3:



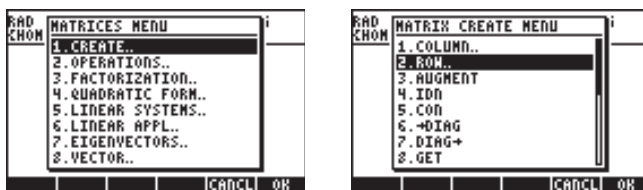
As you can see, the columns that originally occupied positions 2 and 3 have been swapped. Swapping of columns, and of rows (see below), is commonly used when solving systems of linear equations with matrices. Details of these operations will be given in a subsequent Chapter.

## Manipulating matrices by rows

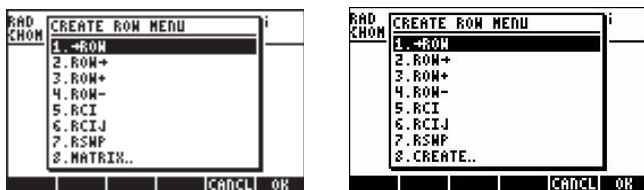
The calculator provides a menu with functions for manipulating matrices by operating in their rows. This menu is available through the MTH/MATRIX/ROW.. sequence: ( MTH ) shown in the figure below with system flag 117 set to CHOOSE boxes:



or through the MATRICES/CREATE/ROW sub-menu:



Both approaches will show the same functions:



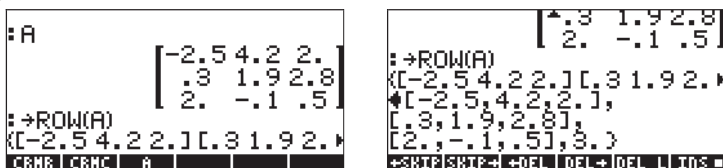
When system flag 117 is set to SOFT menus, the ROW menu is accessible through  $\leftarrow$  MTH  $\leftarrow$  MATRIX  $\leftarrow$  ROW, or through  $\leftarrow$  MATRICES  $\leftarrow$  CREATE  $\leftarrow$  ROW. Both approaches will show the same set of functions:



The operation of these functions is presented below.

## Function $\rightarrow$ ROW

Function  $\rightarrow$ ROW takes as argument a matrix and decomposes it into vectors corresponding to its rows. An application of function  $\rightarrow$ 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  $\nabla$ ).



In RPN mode, you need to list the matrix in the stack, and then activate the function  $\rightarrow$ ROW, i.e.,  $\leftarrow$  →ROW. The figure below shows the RPN stack before and after the application of function  $\rightarrow$ ROW.



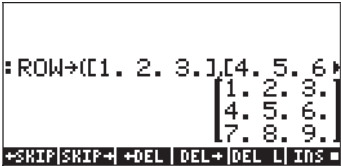
In this result, the first row occupies the highest stack level after decomposition, and stack level 1 is occupied by the number of rows of the original matrix. The

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)
```

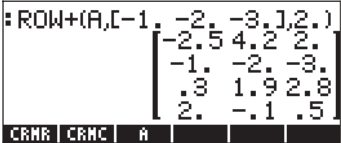


In RPN mode, place the n vectors in stack levels n+1, n, n-1,...,2, and the number n in stack level 1. With this set up, function ROW→ places the vectors as rows in the resulting matrix. The following figure shows the RPN stack before and after using function ROW→.

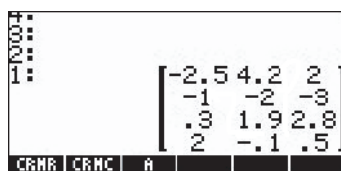
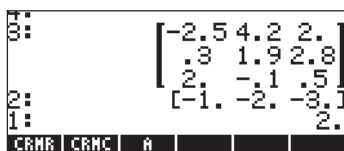


**Function ROW+**

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.,



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+.

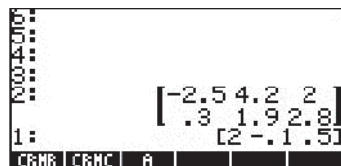


## Function ROW-

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:

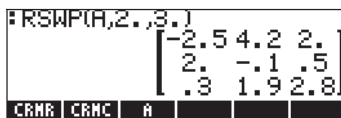


In RPN mode, place the matrix in the stack first, then enter the number representing a row location before applying function ROW-. The following figure shows the RPN stack before and after applying function ROW-.



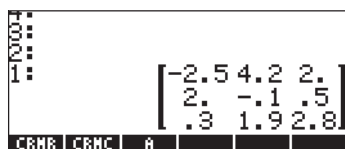
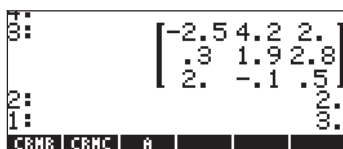
## Function RSWP

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.



In RPN mode, function RSWP 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 RSWP to matrix A in order to swap rows 2 and 3:

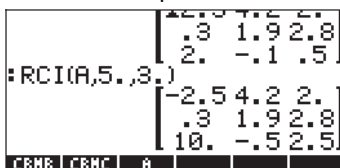




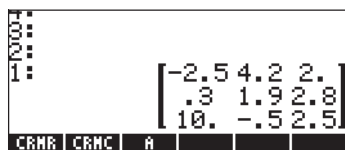
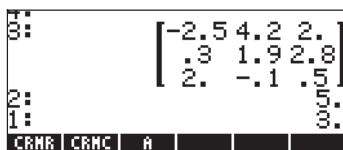
As you can see, the rows that originally occupied positions 2 and 3 have been swapped.

## Function RCI

Function RCI stands for multiplying Row I by a Constant 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.



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.



## 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.)			
$\begin{bmatrix} -2.5 & 4.2 & 2. \\ 3.3 & 1.75 & 3.55 \\ 2. & -.1 & .5 \end{bmatrix}$			
CMNR	CMNC	A	

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:

0:	
4:	$\begin{bmatrix} -2.5 & 4.2 & 2. \\ .3 & 1.9 & 2.8 \\ 2. & -.1 & .5 \end{bmatrix}$
3:	1.5
2:	3.
1:	2.
CMNR	CMNC
A	

0:	
4:	
3:	
2:	$\begin{bmatrix} -2.5 & 4.2 & 2. \\ 3.3 & 1.75 & 3.55 \\ 2. & -.1 & .5 \end{bmatrix}$
1:	
CMNR	CMNC
A	