

Chapter 10

创建和操作矩阵

本章介绍了一些旨在在计算器中创建矩阵并演示矩阵元素操作的示例。

Definitions

矩阵只是具有多个行和列的对象的矩形阵列（例如，数字，代数）。因此，具有 n 行和 m 列的矩阵 \mathbf{A} 将具有 $n \times m$ 个元素。矩阵的通用元素是由索引变量 a_{ij} 表示，对应于行 i 和列 j 。使用这种表示法，我们可以将矩阵 \mathbf{A} 写为 $\mathbf{A} = [a_{ij}]_{n \times m}$ 。完整矩阵如下所示：

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

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 \mathbf{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

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

单位矩阵可写为 $\mathbf{I}_{n \times n} = [\delta_{ij}]$ ，其中 δ_{ij} 是一个称为Kronecker的delta的函数，定义为

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

在堆栈中输入矩阵

在本节中，我们提出了两种在计算器堆栈中输入矩阵的方法：（1）使用 Matrix Writer，以及（2）将矩阵直接输入到堆栈。

使用Matrix Writer

与第9章中讨论的向量的情况一样，可以使用Matrix Writer将矩阵输入到堆栈中。例如，要输入矩阵：

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

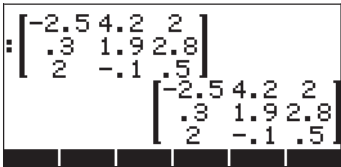
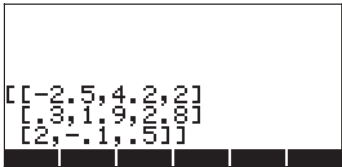
first, start the matrix writer by using \leftarrow MTRW . Make sure that the option $\begin{bmatrix} \square \end{bmatrix} \rightarrow \blacksquare$ is selected. Then use the following keystrokes: 首先，使用启动矩阵编写器。确保选择了GO→选项。然后使用以下按键：

$\left[\begin{matrix} 2 & \cdot & 5 & +/- & ENTER & 4 & \cdot & 2 & ENTER & 2 & ENTER & \downarrow & \leftarrow & \leftarrow & \leftarrow \\ \cdot & 3 & ENTER & / & \cdot & 9 & ENTER & 2 & \cdot & 8 & ENTER \\ 2 & ENTER & \cdot & / & +/- & ENTER & \cdot & 5 & ENTER \end{matrix} \right]$

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

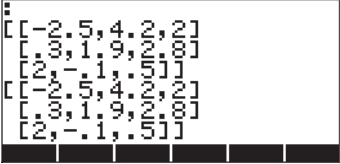


再次按 $\left[\text{ENTER} \right]$ 将矩阵放在堆栈上。接下来，在再次按下 $\left[\text{ENTER} \right]$ 之前和之后显示ALG模式堆栈：



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

如果您选择了教科书显示选项
(使用H@) DISP! 并检
查? 教科书), 矩阵将如上所
示。否则, 显示屏将显示:

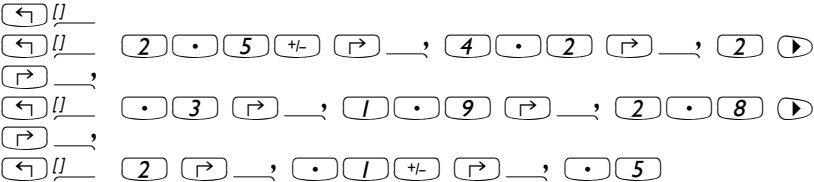


The display in RPN mode will look very similar to these.

Note: 有关矩阵编写器使用的详细信息, 请参见第9章。

直接在矩阵中输入矩阵

通过直接在堆栈中输入以下内容, 可以获得与上述相同的结果:



Thus, to enter a matrix directly into the stack open a set of brackets ((**2nd**) (**F1**)) and enclose each row of the matrix with an additional set of brackets ((**2nd**) (**F1**)). Commas ((**R**), (**.**)) should separate the elements of each row, as well as the brackets between rows. (**Note:**在RPN模式下, 您可以在输入第一组后省略内括号, 因此, 而不是键入, 例如[[1 2 3] [4 5 6] [7 8 9]], 键入[[1 2] 3] 4 5 6 7 8 9)。

对于将来的练习, 让我们将此矩阵保存在名称A. In ALG mode use

(**STO**) (**ALPHA**) (**A**). In RPN mode, use (**.**) (**ALPHA**) (**A**) (**STO**) (**.**).

使用计算器功能创建矩阵

可以使用MTH菜单中MTH / MATRIX / MAKE子菜单中的计算器功能创建一些矩阵 ((**2nd**) (**MTH**)),



或MATRICES / CREATE菜单中  MATRICES 提供的计算器功能创建一些矩阵。:



MTH / MATRIX / MAKE子菜单 (我们称之为MAKE菜单) 包含以下功能:



而MATRICES / CREATE子菜单 (让我们称之为CREATE菜单) 具有以下功能:





从探索这些菜单 (MAKE和CREATE) 可以看出, 它们都具有相同的功能 GET, GETI, PUT, PUTI, SUB, REPL, RDM, RANM, HILBERT, VANDER MONDE, IDN, CON, →DIAG和DIAG→。CREATE菜单包括COLUMN和ROW子菜单, 这些子菜单也可以在MTH / MATRIX菜单下找到。MAKE菜单包括CREIZ菜单不包含的SIZE功能。但是, 基本上, 菜单MAKE和CREATE都为用户提供了相同的功能集。在下面的示例中, 我们将展示如何使用矩阵MAKE菜单访问函数。在本节的最后, 我们提供了一个表格, 其中包含当系统标志117设置为SOFT菜单时, 使用CREATE菜单获得相同功能所需的击键。

如果您已将该系统标志 (标志117) 设置为SOFT菜单, 则MAKE菜单将通过 按键序列提供:

可用功能将显示为软菜单键标签, 如下所示 (按 移至下一组功能):



系统标志117设置为SOFT菜单后, 由 触发的CREATE菜单的功能将显示如下:



在下一节中, 我们将在MAKE和CREATE菜单中介绍矩阵函数的应用。

函数GET和PUT

函数GET, GETI, PUT和PUTI以与列表或向量类似的方式与矩阵一起运行, 即, 您需要提供要GET或PUT的元素的位置。但是, 在列表和向量中, 只需要一个索引来标识元素, 在矩阵中, 我们需要一个包含两个索引{row, column}的列表来标识矩阵元素。下面是使用GET和PUT的例子。

让我们将上面存储的矩阵用于变量A来演示使用GET和PUT功能。例如, 要在ALG模式下从矩阵A中提取元素a₂₃, 可以按如下方式执行:

```
:GET(A,{2 3})
2.8
:A(2,3)
2.8
GET | GETI | PUT | PUTI | SUB | REPL
```

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 **ENTER** **3** **ENTER** GET, or by using A(2,3) **ENTER**.

Suppose that we want to place the value ' π ' into element a₃₁ of the matrix. We can use function PUT for that purpose, e.g.,

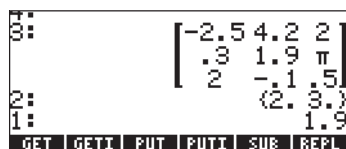
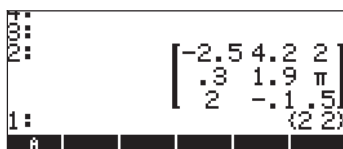
假设我们想要将值 π 放入矩阵的元素a₃₁中。我们可以为此目的使用函数PUT, 例如,

```
: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: **VAR** **ENTER** **{3,1}** **ENTER** **π** PUT. Alternatively, in RPN mode you can use: **π** **A(2,3)** **ENTER** **STOP**. To see the contents of variable A after this operation, use **ENTER**.

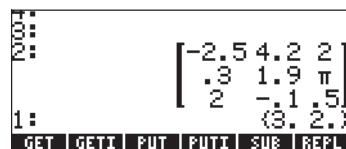
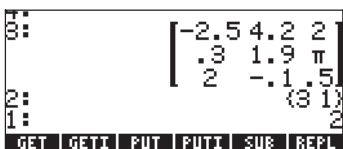
函数GETI和PUTI

函数PUTI和GETI用于UserRPL程序, 因为它们跟踪重复应用PUT和GET功能的索引。矩阵中的索引列表首先按列排列。为了说明其用途, 我们建议采用RPN模式进行以下练习: **ENTER** **{2,2}** **ENTER** GETI. 在应用GETI功能之前和之后显示RPN堆栈的屏幕截图如下所示:



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. 请注意，通过将原始引用的列索引增加1（即，从{2,2}到{2,3}），屏幕为随后的

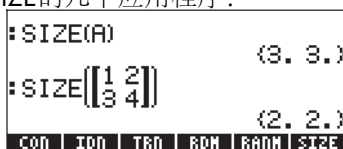
GETI或GET应用做好准备，同时显示提取的值 $A(2,2) = 1.9$ ，即堆栈级别1. Now, suppose that you want to insert the value 2 in element {3 1} using PUTI. Still in RPN mode, try the following keystrokes: \leftarrow \leftarrow {3 1} \rightarrow ENTER \rightarrow 2 \rightarrow ENTER PUTI. The screen shots below show the RPN stack before and after the application of function PUTI:



在这种情况下，2被替换为位置{3 1}，即现在 $A(3,1) = 2$ ，并且索引列表增加1（按列第一），即从{3,1}增加到{3,2}。矩阵在级别2中，递增的索引列表在级别1中。

Function SIZE

函数SIZE提供了一个列表，显示了堆栈级别1中矩阵的行数和列数。以下屏幕显示了ALG模式下函数SIZE的几个应用程序：



In RPN mode, these exercises are performed by using \leftarrow SIZE, and \leftarrow 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):

函数TRN用于产生矩阵的反式共轭，即转置（TRAN），然后是其复共轭（CONJ）。例如，以下屏幕截图显示了原始矩阵

```

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

:A-i:2:A
      [-2.5--2.5i:2 4.2-4.2i:2 2-2i:2]
      [.3-.3i:2 1.9-1.9i:2 π-πi:2]
      [2-2i:2 -.1--.1i:2 .5-.5i:2]

CON IDN TRN RDM RANM SIZE

```

```


:A-i:2:A
      [-2.5--2.5i:2 4.2-4.2i:2 2-2i:2]
      [.3-.3i:2 1.9-1.9i:2 π-πi:2]
      [2-2i:2 -.1--.1i:2 .5-.5i:2]

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

CON IDN TRN RDM RANM SIZE

```

如果参数是实矩阵，则TRN仅产生实矩阵的转置。例如，尝试TRN (A) ，并将其与TRAN (A) 进行比较。

在RPN模式中，矩阵A的反式共轭通过使用来计算  TRN.

Note: 计算器还包括MATRICES / OPERATIONS子菜单中的Function TRAN:



For example, in ALG mode:

```

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

A

```

Function CON

该函数将两个元素的列表作为参数，对应于要生成的矩阵的行数和列数，以及常量值。函数CON生成具有常量元素的矩阵。例如，在ALG模式下，以下命令创建一个4×3矩阵，其元素全部等于-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|INS+

```

在RPN模式下，这是通过使用来完成的 {4,3} **ENTER** **/** **.** **5** **+/-** **ENTER** **CON**.

Function IDN

函数IDN (IDeNtity矩阵) 根据其大小创建单位矩阵。回想一下，单位矩阵必须是方阵，因此，只需要一个值就可以完全描述它。例如，要在ALG模式下创建4×4单位矩阵，请使用：

```

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

```

您还可以使用现有的方阵作为函数IDN的参数，例如，

```

: IDN(A)
  [1 0 0]
  [0 1 0]
  [0 0 1]
CON | IDN | TRN | RDM | RANM | SIZE

```

得到的单位矩阵将具有与参数矩阵相同的维度。请注意，尝试使用矩形（即非方形）矩阵作为IDN的参数将产生错误。

在RPN模式下，上面显示的两个练习是使用以下方法创建的： **4** **ENTER**

IDN and  IDN.

Function RDM

函数RDM (Re-DiMensioning) 用于将向量和矩阵重写为矩阵和向量。函数的输入包括原始向量或矩阵，后跟单个数字列表（如果转换为向量）或两个数字（如果转换为矩阵）。在前一种情况下，数字表示向量的维度，在后者中表示矩阵的行数和列数。以下示例说明了函数RDM的用法：

将向量重新标注为矩阵

以下示例显示如何在ALG模式下将6个元素的向量重新标注为2行3列的矩阵：

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

在RPN模式下，我们可以使用 [1,2,3,4,5,6] **ENTER** {2,3} **ENTER** RDM 来生成上面显示的矩阵。

将矩阵重新标注为另一个矩阵

在ALG模式下，我们现在使用上面创建的矩阵并将其重新标注为3行和2列的矩阵：

```
: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.

将矩阵重新标注为向量

为了将矩阵重新标注为向量，我们使用矩阵作为参数，后跟包含矩阵中元素数量的列表。例如，要将前一个示例中的矩阵转换为长度为6的向量，请在ALG模式下使用：

```
: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
```

如果使用RPN模式，我们假设矩阵在堆栈中并使用{6} **ENTER** RDM。

Note:函数RDM提供了一种更直接有效的方法，可以将列表转换为数组，反之亦然，而不是第9章末尾提供的方法。

Function RANM

函数RANM (RANdom Matrix) 将生成具有随机整数元素的矩阵，给定具有行数和列数的列表（即，矩阵的维度）。例如，在ALG模式下，使用相同的命令产生两个具有随机元素的不同2×3矩阵，即：

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} **ENTER** RANM.

显然，您在计算器中得到的结果肯定会与上面显示的结果不同。产生的随机数是均匀分布在[-10,10]范围内的整数，即，这21个数中的每一个具有相同的被选择概率。函数RANM可用于生成任何大小的矩阵，以说明矩阵运算或矩阵函数的应用。

Function SUB

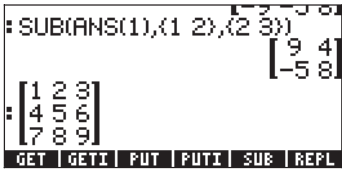
功能SUB从现有矩阵中提取子矩阵，前提是指示子矩阵的初始和最终位置。例如，如果我们想要从最后一个结果中提取元素a12, a13, a22和a23，作为2×2子矩阵，在ALG模式下，使用：

```
: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
```

在RPN模式下，假设原始的2x3矩阵已经在堆栈中，请使用
In RPN mode, assuming that the original 2x3 matrix is already in the stack, use
(1,2) **ENTER** (2,3) **ENTER** SUB.

函数REPL将子矩阵替换或插入较大的子矩阵。此功能的输入是将进行替换的
矩阵，替换开始的位置以及要插入的矩阵。例如，保留我们从前一个示例继

Function REPL 承的矩阵，输入矩阵: [[1,2,3], [4,5,6], [7,8,9]]。
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
FNS(2), the 2x2 matrix, into the 3x3 matrix currently located in FNS(1),
starting at position (2,2):



如果在RPN模式下工作，假设2x2矩阵最初位于堆栈中，我们按如下方式进行：

`[[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 采用尺寸为n×n的方阵的主对角线，并创建包含主对角线元素的
维数n的向量。例如，对于前一练习剩余的矩阵，我们可以使用以下方法
提取其主对角线：



在RPN模式下，如果堆栈中有3×3矩阵，我们只需激活函数→DIAG即可获得与上述相同的结果。

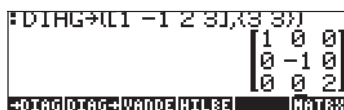
In RPN mode, with the 3×3 matrix in the stack, we simply have to activate function →DIAG to obtain the same result as above.

Function DIAG→

Function DIAG→ 获取矢量和矩阵维列表{行，列}，并创建一个对角矩阵，主对角线用适当的矢量元素替换。例如，命令

DIAG→([1,-1,2,3],[3,3])

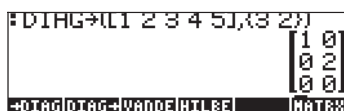
使用向量参数的前3个元素生成对角矩阵：



```
DIAG→([1 -1 2 3],[3 3])
      1 0 0
      0 -1 0
      0 0 2
→DIAG|DIAG→|VANDERMONDE| MATRX
```

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 4
→DIAG|DIAG→|VANDERMONDE| MATRX
```

In RPN mode, use [1,2,3,4,5] (ENTER) (3,2) (ENTER) DIAG→ .

在这种情况下，使用尽可能多的元素作为主对角元素来创建3×2矩阵，形成向量[1,2,3,4,5]。对于矩形矩阵，主对角线从位置 (1,1) 开始并移动到位置 (2,2)，(3,3) 等，直到行或列的数量耗尽。在这种情况下，列数 (2) 在行数 (3) 之前耗尽，因此主对角线仅包括位置 (1,1) 和 (2,2) 中的元素。因此，仅需要矢量的前两个元素来形成主对角线。

Function VANDERMONDE

函数VANDERMONDE根据给定的输入数据列表生成维度为n的Vandermonde矩阵。当然，维度n是列表的长度。如果输入列表由对象{x1, x2, ... xn}组成，则计算器中的Vandermonde矩阵是由以下元素组成的矩阵：

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix}$$

例如，ALG模式中的以下命令用于列表{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
+DIAG|DIAG+VANDERMONDE|HILBERT|MATRIX

```

In RPN mode, enter { 1, 2, 3, 4 } **ENTER** VANDERMONDE.

Function HILBERT

函数HILBERT创建对应于维度n的希尔伯特矩阵。根据定义，n×n希尔伯特矩阵是 $\mathbf{H}_n = [h_{jk}]_{n \times n}$, so that

$$h_{jk} = \frac{1}{j+k-1}$$

Hilbert矩阵通过线性平方法在数值曲线拟合中得到应用。

从多个列表中构建矩阵的程序

在本节中，我们提供了一些UserRPL程序，用于从多个对象列表中构建矩阵。列表可以表示矩阵的列(program **COLS**) 或矩阵的行(program **ROWS**)。在计算器设置为RPN模式的情况下输入程序，并且为设置为SOFT菜单的系统标志117给出击键指令。本节旨在让您练习在计算器中访问编程功能。下面列出了程序，在左侧显示了输入程序步骤所需的击键，在右侧显示了字符

在右侧显示了在执行这些击键时在显示屏中输入的字符。 首先，我们介绍了生成CRMC程序所需的步骤。

列表表示矩阵的列

The program **LIST** 允许您将每个p个元素的n个列表中的p×n矩阵（即，p行，n列）放在一起。 要创建程序，请输入以下按键：

Keystroke sequence:

⏮ ⏭
⏮ PRG **LIST**
⏮ → (SPC) (ALPHA) ⏮ (N)
⏮ ⏭
/ ⏮ PRG **LIST**
⏮ PRG **BRCH** **FOR** **FOR**
(ALPHA) ⏮ (I)
⏮ PRG **TYPE** **OBJ** →
→ **ARRY**
⏮ PRG **BRCH** **IF** **IF**
(ALPHA) ⏮ (I) (SPC)
(ALPHA) ⏮ (N)
⏮ PRG **TEST** **<**
⏮ PRG **BRCH** **IF** **THEN**
(ALPHA) ⏮ (I) (SPC) (I) (+)
⏮ PRG **STACK** **NXT** **ROLL**
⏮ PRG **BRCH** **IF** **END**
⏮ PRG **BRCH** **FOR** **NEXT**
⏮ PRG **BRCH** **IF** **IF**
(ALPHA) ⏮ (N) (SPC) (I)
⏮ PRG **TEST** **>**
⏮ PRG **BRCH** **IF** **THEN**
/ (SPC)
(ALPHA) ⏮ (N) (SPC) (I) (-)
⏮ PRG **BRCH** **FOR** **FOR**
(ALPHA) ⏮ (I) (SPC)
(ALPHA) ⏮ (I) (SPC) (I) (+)
⏮ PRG **STACK** **NXT** **ROLL**
⏮ PRG **BRCH** **FOR** **NEXT**
⏮ PRG **BRCH** **IF** **END**

Produces:

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

ALPHA \leftarrow (N) SPC
 \leftarrow MTH \leftarrow \rightarrow
 ENTER

n
 COL \rightarrow
 Program is displayed in level 1

To save the program:

() ALPHA ALPHA (C) (R) (M) (C) ALPHA (STOP)

Note: 如果将此程序保存在HOME目录中，则可以从您使用的任何其他子目录中获取该程序。

To see the contents of the program use VAR \rightarrow \rightarrow \rightarrow . The program listing is the following:

```

« DUP  $\rightarrow$  n « 1 SWAP FOR j OBJ $\rightarrow$   $\rightarrow$ 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 $\rightarrow$  » »
  
```

要使用此程序，在RPN模式下，按照您希望它们作为矩阵列的顺序输入n个列表，输入值n，然后按 \rightarrow 。例如，尝试以下练习：

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

以下屏幕截图显示了运行程序 \rightarrow 之前和之后的 \rightarrow 堆栈：

7:	
6:	(1 2 3 4)
5:	(1 4 9 16)
4:	(1 8 27 64)
3:	3
2:	
1:	

CRMC | A | | | |

1:	1 1 1
	2 4 8
	3 9 27
	4 16 64

CRMC | A | | | |

要在ALG模式下使用该程序，请按 \rightarrow 然后按一组括号(\leftarrow)。在括号内键入表示矩阵列的数据列表，以逗号分隔，最后是逗号和列数。该命令应如下所示：

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

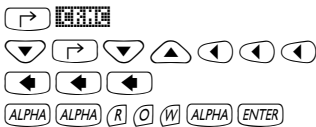
显示程序CRMC执行的ALG屏幕如下所示：

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

HYP | ACOS2 | ASIN2 | ASIN2 | ATAN2 | HALFT

列表表示矩阵的行

当输入列表将成为结果矩阵的行时，可以轻松修改先前的程序以创建矩阵。要执行的唯一更改是在程序列表中更改COL→for ROW→。要执行此更改，请使用：



List program CRMC in stack

Move to end of program

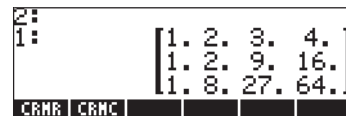
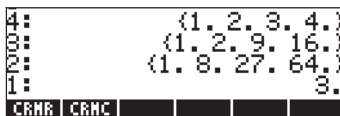
Delete COL

Type in ROW, enter program

To store the program use:


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

The following screen shots show the RPN stack before and after running program `0010`:



这些程序可用于统计应用，特别是创建统计矩阵 Σ DAT。后面的章节将介绍这些程序的使用示例。

按列操作矩阵

计算器提供了一个菜单，其中包含通过在列中操作来操作矩阵的功能。此菜单可通过MTH / MATRIX / COL ..序列：( MTH) 显示，如下图所示，系统标志117设置为CHOOSE框：



或通过MATRICES / CREATE / COLUMN子菜单:



两种方法都将显示相同的功能:



当系统标志117设置为SOFT菜单时, COL菜单可通过

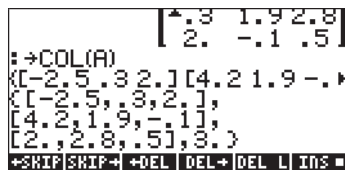
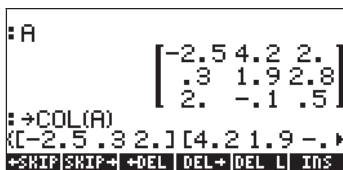
\leftarrow MTH \leftarrow MATRIX \leftarrow COL, 或 \leftarrow MATRICES \leftarrow CREATE \leftarrow COL. 两种方法都将显示相同的功能集:



这些功能的操作如下所示。

Function →COL

Function →COL 将矩阵作为参数, 并将其分解为与其列对应的向量。ALG模式下的函数? COL的应用如下所示。使用的矩阵早先存储在变量A中。矩阵如左图所示。右图显示了按列分解的矩阵。要查看完整结果, 请使用行编辑器(triggered by pressing ∇)。



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.


```

:COL+(A,[-1. -2. -3.],2.)
[-2.5 -1.4 2. 2.]
[.3 -2. 1.9 2.8]
[2. -3. -.1 .5]
CRNR | CRMC | A

```

在RPN模式下，在应用函数COL +之前，首先输入矩阵，然后输入矢量和列号。 下图显示了应用函数COL +之前和之后的RPN堆栈。

```

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

```

```

4:
3:
2:
1:
[-2.5 -1.4 2. 2.]
[.3 -2. 1.9 2.8]
[2. -3. -.1 .5]
CRNR | CRMC | A

```

Function COL-

函数COL-将矩阵和整数表作为参数，表示矩阵中列的位置。 函数返回原始矩阵减去一列，以及显示为向量的提取列。 以下是使用存储在A中的矩阵的ALG模式示例：

```

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

```

在RPN模式下，首先将矩阵放入堆栈，然后在应用函数COL-之前输入表示列位置的数字。 下图显示了应用函数COL-之前和之后的RPN堆栈。

```

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

```

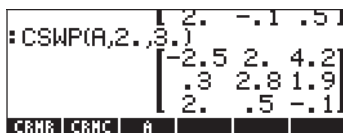
```

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

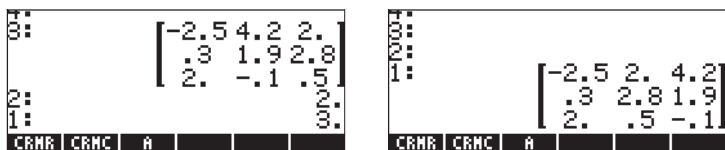
```

Function CSWP

函数CSWP（列SWaP）将两个索引（例如，i和j）（表示矩阵中的两个不同列）和矩阵作为参数，并生成一个新的矩阵，其中列i和j交换。 以下示例在ALG模式下显示此功能的应用程序。 我们使用存储在变量A中的矩阵作为示例。 首先列出该矩阵。



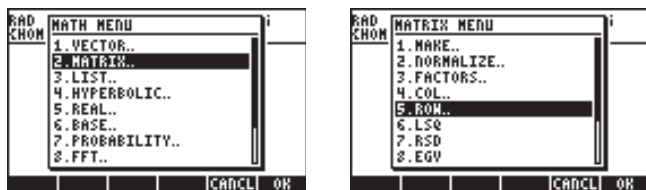
在RPN模式下，函数CSWP允许您交换堆栈级别3中列出的矩阵的列，其索引在堆栈级别1和2中列出。例如，下图显示了将函数CSWP应用于矩阵A之前和之后的RPN堆栈 为了交换第2列和第3列：



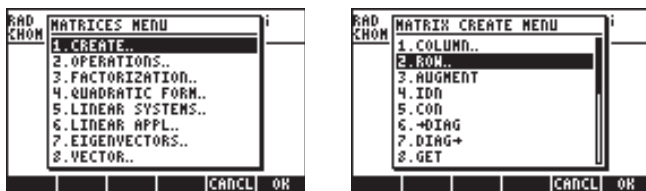
如您所见，最初占据位置2和3的列已被交换。当求解具有矩阵的线性方程组时，通常使用列和行的交换（见下文）。这些操作的细节将在随后的章节中给出。

按行操作矩阵

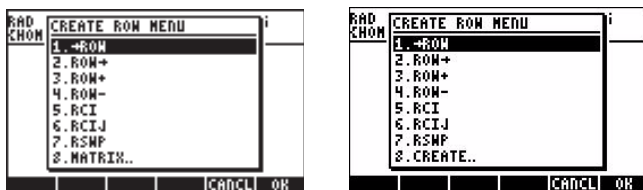
计算器提供了一个菜单，其中包含通过操作行来操作矩阵的功能。此菜单可通过MTH / MATRIX / ROW ..序列获得：(MTH) 如下图所示，系统标志117设置为CHOOSE框：



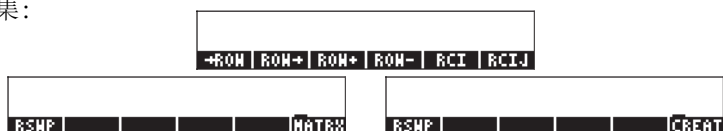
或通过MATRICES / CREATE / ROW子菜单：



两种方法都将显示相同的功能：



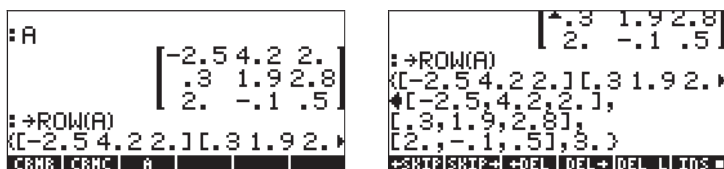
当系统标志117设置为SOFT菜单时，可以通过 \leftarrow MTH \leftarrow MATRIX \leftarrow ROW , or through \leftarrow MATRICES \leftarrow EDIT \leftarrow ROW 访问ROW菜单。两种方法都将显示相同的功能集：



这些功能的操作如下所示。

Function →ROW

函数→ROW将矩阵作为参数，并将其分解为与其行对应的向量。ALG模式下的函数→ROW的应用如下所示。使用的矩阵早先存储在变量A中。矩阵如左图所示。右图显示了按行分解的矩阵。要查看完整结果，请使用行编辑器 (triggered by pressing ∇) .



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

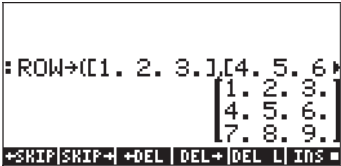


在该结果中，第一行在分解之后占据最高堆栈级别，并且堆栈级别1被原始矩阵的行数占据。矩阵不会在分解中存活，即它在堆栈中不再可用。

Function ROW→

函数ROW→具有与函数→ROW相反的效果，即，给定n个相同长度的向量，并且数n，函数ROW→ 通过将输入向量作为结果矩阵的行来构建矩阵。 以下是ALG模式的示例。 使用的命令是：

```
ROW→([1,2,3],[4,5,6],[7,8,9],3)
```

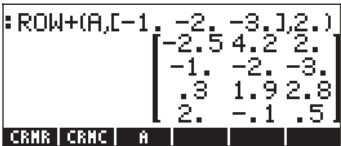


在RPN模式下，将n个向量放在堆栈级别n + 1，n，n-1，...，2和堆栈级别1中的数字n中。通过此设置，函数ROW→ 将矢量作为行放在结果矩阵中。 下图显示了使用函数ROW→之前和之后的RPN堆栈。



Function ROW+

函数ROW +将矩阵，与矩阵中行数相同的向量作为参数，以及表示行位置的整数n。 函数ROW +将向量插入矩阵的行n中。 例如，在ALG模式下，我们将使用向量[-1， -2， -3]在矩阵A中插入第二行，即



在RPN模式下，在应用函数ROW +之前，首先输入矩阵，然后输入向量和行号。 下图显示了应用函数ROW +之前和之后的RPN堆栈。

3:					
2:					
1:					
CRNR CRNC A					

3:					
2:					
1:					
CRNR CRNC A					

Function ROW-

函数ROW-将矩阵和整数表作为参数，表示矩阵中行的位置。该函数返回原始矩阵，减去一行，以及显示为向量的提取行。以下是使用存储在A中的矩阵的ALG模式示例：

:ROW-(A,3.)					
[[-2.5 4.2 2.]					
[.3 1.9 2.8] [2. -.1 .5]					
CRNR CRNC A					

在RPN模式下，首先将矩阵放在堆栈中，然后在应用函数ROW-之前输入表示行位置的数字。下图显示了应用函数ROW-之前和之后的RPN堆栈。

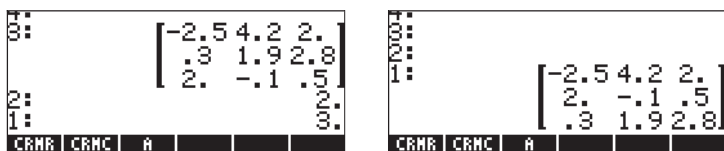
5:					
4:					
3:					
2:					
1:					
CRNR CRNC A					

Function RSWP

函数RSWP（行SWaP）将两个索引（例如，i和j）（表示矩阵中的两个不同的行）和矩阵作为参数，并生成一个新的矩阵，其中行i和j交换。以下示例在ALG模式下显示此功能的应用程序。我们使用存储在变量A中的矩阵作为示例。首先列出该矩阵。

:RSWP(A,2.,3.)					
[[-2.5 4.2 2.]					
[2. -.1 .5]					
.3 1.9 2.8]					
CRNR CRNC A					

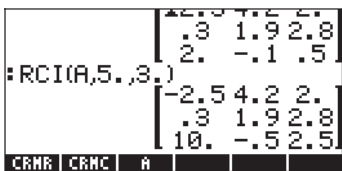
在RPN模式下，函数RSWP允许您交换堆栈级别3中列出的矩阵的行，其索引在堆栈级别1和2中列出。例如，下图显示了将函数RSWP应用于矩阵A之前和之后的RPN堆栈 为了交换第2行和第3行：



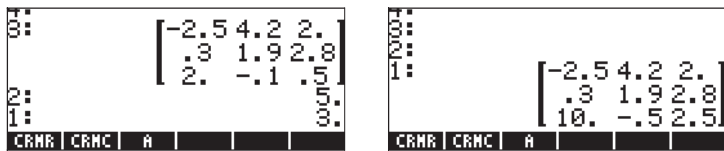
如您所见，最初占据位置2和3的行已被交换。

Function RCI

函数RCI表示将第I行乘以常量值，并将结果行替换为相同的位置。以ALG模式编写的以下示例获取存储在A中的矩阵，并将常量值5乘以行号3，用此产品替换该行。



在RPN模式下完成的相同练习如下图所示。左侧图显示了堆栈级别3,2和1中矩阵，因子和行号的设置。右侧图显示了激活RCI功能后的结果矩阵。



Function RCIJ

函数RCIJ代表“取第I行并将其乘以常数C，然后将该行乘以行J添加，将行J替换为行J。”这种类型的行操作在高斯或高斯过程中非常常见。消除约旦（关于这一程序的更多细节将在随后的章节中介绍）。该函数的参数是：（1）矩阵，（2）常数值，（3）要乘以（2）中的常数的行，以及（4）要由结果和替换的行 如上所述。例如，取存储在变量A中的矩阵，我们将第3列乘以1.5，并将其添加到第2列。以下示例在ALG模式下执行：

```

RCIJ(A,1.5,3,2.)
      [-2.5 4.2 2.]
      [3.3 1.75 3.55]
      [2. -1. .5]
CRNR CRNC A

```

在RPN模式下，首先输入矩阵，然后输入常量值，然后输入要乘以常数值
的行，最后输入要替换的行。 下图显示了在与上面显示的ALG示例相同的条件
下应用函数RCIJ之前和之后的RPN堆栈：

```

0:
4:      [-2.5 4.2 2.]
      [3.3 1.9 2.8]
      [2. -1. .5]
3:
2:
1:      1.5
      3.5
      2.
CRNR CRNC A

```

```

0:
4:      [-2.5 4.2 2.]
      [3.3 1.75 3.55]
      [2. -1. .5]
3:
2:
1:
CRNR CRNC A

```