

# Chapter 11

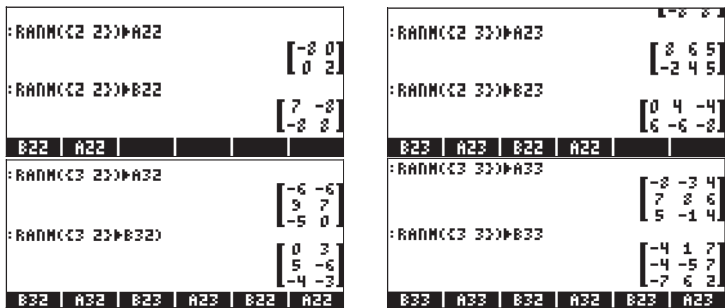
## 矩阵运算和线性代数

在第10章中，我们介绍了矩阵的概念，并提供了许多用于输入，创建或操作矩阵的函数。在本章中，我们提出矩阵运算的例子和线性代数问题的应用。

### 使用矩阵的操作

与其他数学对象一样，可以添加和减去矩阵。 它们可以乘以标量，或者相互之间。 他们也可以提升到真正的力量。 线性代数应用的一个重要操作是矩阵的逆矩阵。 接下来介绍这些操作的细节。




为了说明操作，我们将创建一些我们将存储在变量中的矩阵。 矩阵的通用名称是 $A_{ij}$ 和 $B_{ij}$ ，其中 $i$ 表示行数， $j$ 表示矩阵的列数。 要使用的矩阵是通过使用函数RANM（随机矩阵）生成的。 如果您在计算器中尝试此练习，您将得到与此处列出的矩阵不同的矩阵，除非您将它们完全按照下图所示存储到计算器中。 以下是在ALG模式下创建的矩阵A22，B22，A23，B23，A32，B32，A33和B33：

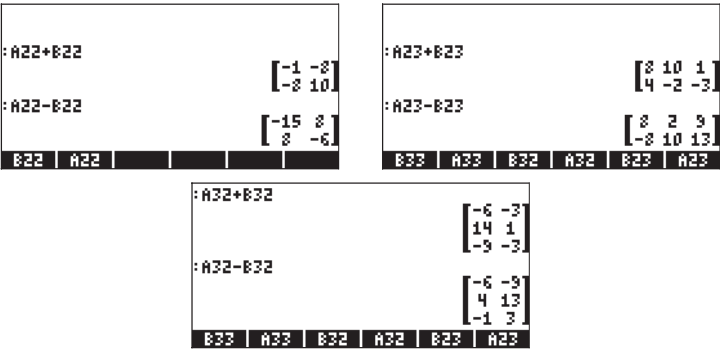


在RPN模式下，要遵循的步骤是：



















(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▶)

# 加减

考虑一对矩阵  $\mathbf{A} = [a_{ij}]_{m \times n}$  and  $\mathbf{B} = [b_{ij}]_{m \times n}$ . 只有当它们具有相同数量的行和列时，才能添加和减去这两个矩阵。得到的矩阵，  $\mathbf{C} = \mathbf{A} \pm \mathbf{B} = [c_{ij}]_{m \times n}$  has elements  $c_{ij} = a_{ij} \pm b_{ij}$ . ALG模式的一些示例使用上面存储的矩阵显示如下 (e.g.,   )



在RPN模式下，要遵循的步骤是：

$A_{22}$    $B_{22}$         $A_{22}$    $B_{22}$     
 $A_{23}$    $B_{23}$         $A_{23}$    $B_{23}$     
 $A_{32}$    $B_{32}$         $A_{32}$    $B_{32}$   

将ALG示例转换为RPN非常简单，如此处所示。     矩阵运算的其余示例仅在ALG模式下执行。

# 乘法

有许多涉及矩阵的乘法运算。 这些将在下面描述。

## 用标量乘法

Multiplication of the matrix  $\mathbf{A} = [a_{ij}]_{m \times n}$  by a scalar  $k$  results in the matrix  $\mathbf{C} = k\mathbf{A} = [c_{ij}]_{m \times n} = [ka_{ij}]_{m \times n}$ . 特别地，矩阵的负值由操作  $-\mathbf{A} = (-1)\mathbf{A} = [-a_{ij}]_{m \times n}$  定义。 下面示出了通过标量乘以矩阵的一些示例。



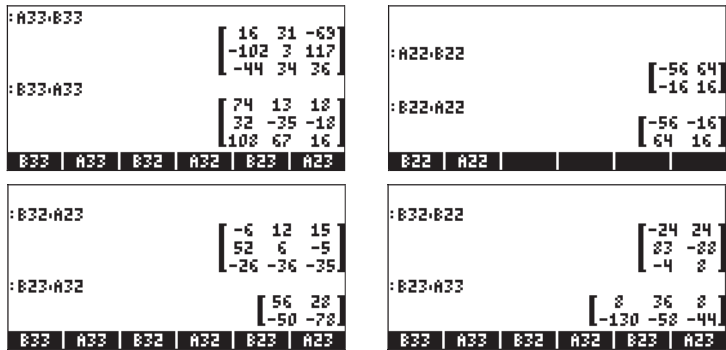
矩阵乘法

矩阵乘法由  $\mathbf{C}_{m \times n} = \mathbf{A}_{m \times p} \cdot \mathbf{B}_{p \times n}$  定义，其中  $\mathbf{A} = [a_{ij}]_{m \times p}$ ,  $\mathbf{B} = [b_{ij}]_{p \times n}$ , and  $\mathbf{C} = [c_{ij}]_{m \times n}$ . 请注意，只有当第一个操作数中的列数等于第二个操作数的行数时，才能进行矩阵乘法。产品中的通用术语  $c_{ij}$  定义为

$$c_{ij} = \sum_{k=1}^p a_{ik} \cdot b_{kj}, \text{ for } i = 1, 2, \dots, m; j = 1, 2, \dots, n.$$

这与产品C的第i行和第j列中的元素C相同，是将A的第i行与B的第j列相乘得出的，并将产品加在一起。矩阵乘法不是可交换的，即通常  $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$ 。此外，其中一个甚至可能不存在。

以下屏幕截图显示了我们之前存储的矩阵的乘法结果：



在前一部分中引入的矩阵向量乘法可以被认为矩阵  $m \times n$  与矩阵  $n \times 1$  (即，列向量) 的乘积，得到  $m \times 1$  矩阵 (即，另一个向量)。要验证此断言，请检查上一节中提供的示例。因此，第9章中定义的向量基本上是由于矩阵乘法的列向量。

如果矢量是行矢量，即  $1 \times m$  矩阵，则乘以矩阵  $m \times n$  的矢量与矩阵的乘积是可能的，产生  $1 \times n$  矩阵 (另一行矢量)。要使计算器识别行向量，必须使用双括号输入：

```

: B32
      [ 0 3 ]
      [ 5 -6 ]
      [-4 -3 ]
: [1 3 6]·ANS(1)
      [-9 -33]
B32 | 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

```

## 逐项乘法

通过使用函数HADAMARD，可以对相同维度的两个矩阵进行逐项乘法。结果当然是另一个相同维度的矩阵。此功能可通过功能目录 ( $\rightarrow$  CAT ), 或 MATRICES / OPERATIONS子菜单 ( $\leftarrow$  MATRICES ) 获得。

接下来介绍HADAMARD功能的应用：

```

: 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

```

## 将矩阵提升到实际功率

只要功率是整数或实数而没有小数部分，您就可以将矩阵提升到任何功率。以下示例显示了将之前创建的矩阵B22提升到5的幂的结果：

```

: B22
      [ 7 -8 ]
      [-8 8 ]
: B22^5
      [ 421543 -448712 ]
      [-448712 477632 ]
EDIT | VIEW | RCL | STOP | PURGE | CLEAR

```

您也可以将矩阵提升为幂而无需先将其存储为变量：

在代数模式中，击键是：[输入或选择矩阵]**Q**[输入功率]**( $y^x$ )****(ENTER)**.在RPN模式下，击键是：[输入或选择矩阵]**( $\text{SPC}$ )** [输入功率]**( $y^x$ )****(ENTER)**.

矩阵可以提升为负幂。在这种情况下，结果相当于 $1/[\text{矩阵}]^{\text{ABS (功率)}}$ 。

## 单位矩阵

在第9章中，我们引入单位矩阵作为矩阵  $\mathbf{I} = [\delta_{ij}]_{n \times n}$ , where  $\delta_{ij}$  是Kronecker的delta函数。可以通过使用第9章中描述的函数IDN来获得标识矩阵。单位矩阵具有  $\mathbf{A} \cdot \mathbf{I} = \mathbf{I} \cdot \mathbf{A} = \mathbf{A}$  的属性。为了验证该属性，我们使用先前存储的矩阵来呈现以下示例：

## 逆矩阵

方矩阵  $\mathbf{A}$  的倒数是矩阵  $\mathbf{A}^{-1}$  使得  $\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{I}$ , 其中  $\mathbf{I}$  是与  $\mathbf{A}$  相同维度的单位矩阵。矩阵的逆矩阵在 计算器使用反函数INV (i.e., the **( $1/x$ )** key). 下面给出一个先前存储的矩阵的逆的示例：

:INV(A33)					
			-19	-4	25
			249	249	249
			-1	26	-38
			249	249	249
			47	23	43
			498	498	498
B33	A33	B32	A32	B23	A23

要验证逆矩阵的属性，请考虑以下乘法：

:A33*INV(A33)					
			1	0	0
			0	1	0
			0	0	1
:INV(B33)*B33					
			1	0	0
			0	1	0
			0	0	1
B33	A33	B32	A32	B23	A23

:B22*INV(B22)					
			0	1	0
			0	0	1
:A22*INV(A22)					
			1	0	
			0	1	
			1	0	
			0	1	
B22	A22				

## 表征矩阵（矩阵**NORM**菜单）

通过击键序列  $\leftarrow$  MTH (system flag 117 set to CHOOSE boxes):

RAD		MATH MENU			
CHOM		1. VECTOR..			
		2. MATRIX..			
		3. LIST..			
		4. HYPERBOLIC..			
		5. REAL..			
		6. BASE..			
		7. PROBABILITY..			
		8. FFT..			
				CANCL	OK

RAD		MATRIX MENU			
CHOM		1. MAKE..			
		2. NORMALIZE..			
		3. FACTORS..			
		4. COL..			
		5. ROW..			
		6. LSQ			
		7. RSD			
		8. EGV			
				CANCL	OK

This menu contains the following functions:

RAD		MATRIX NORM MENU			
CHOM		1. ABS			
		2. SNRM			
		3. RNRM			
		4. CNRM			
		5. SRAD			
		6. COND			
		7. RANK			
		8. DET			
				CANCL	OK

RAD		MATRIX NORM MENU			
CHOM		4. CNRM			
		5. SRAD			
		6. COND			
		7. RANK			
		8. DET			
		9. TRACE			
		10. TRAN			
		11. MATRIX..			
				CANCL	OK

接下来描述这些功能。因为许多这些函数使用矩阵理论的概念，例如奇异值，等级等，所以我们将包括与函数描述混合的这些概念的简短描述。

### Function ABS

函数ABS计算所谓的矩阵的Frobenius范数。对于矩阵 $A = [a_{ij}]_{m \times n}$ ，矩阵的Frobenius范数定义为

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m a_{ij}^2}$$

如果在行向量或列向量中考虑矩阵，那么Frobenius范数 $\|A\|_F$ 就是向量的大小。功能ABS可直接在键盘上用作  $\leftarrow ABS$ 。

在ALG模式下尝试以下练习（使用前面存储的矩阵进行矩阵运算）：

:IA22I	
:IA23I	$2\sqrt{17}$
:IB32I	$\sqrt{170}$
	$\sqrt{95}$
A	X
B32	A33
B32	A32

:IB33I	
:IA33I	$7\sqrt{5}$
:IA32I	$2\sqrt{70}$
	$\sqrt{227}$
A	X
B32	A33
B32	A32

### Function SNRM

函数SNRM计算矩阵的Spectral NoRM，矩阵被定义为矩阵的最大奇异值，也称为矩阵的欧几里德范数。例如，

:SNRM(A22)	8.
:SNRM(A32)	14.7146399549
:SNRM(A33)	14.1867419471
A	X
B32	A33
B32	A32



## 奇异值分解

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  $\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \cdot \mathbf{S}_{m \times n} \cdot \mathbf{V}_{n \times n}^T$ , where  $\mathbf{U}$  and  $\mathbf{V}$  are orthogonal matrices, and  $\mathbf{S}$  is a diagonal matrix. The diagonal elements of  $\mathbf{S}$  are called the singular values of  $\mathbf{A}$  and are usually ordered so that  $s_i \geq s_{i+1}$ , for  $i = 1, 2, \dots, n-1$ . The columns  $[\mathbf{u}_i]$  of  $\mathbf{U}$  and  $[\mathbf{v}_i]$  of  $\mathbf{V}$  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).

可以通过计算非奇异值的数量从其SVD确定矩阵的秩。SVD的示例将在后续部分中介绍。

函数RNRm返回矩阵的行NoRM，而函数CNRm返回矩阵的NoRM列。  
例子，

### 矩阵的行范数和列范数

通过获取每行中所有元素的绝对值的总和，然后选择这些总和的最大值来计算矩阵的行范数。通过获取每列中所有元素的绝对值的总和，然后选择这些总和的最大值来计算矩阵的列范数。

# Function SRAD

函数SRAD确定矩阵的 Spectral RADIUS，定义为其特征值绝对值的最大值。 例如，

```
:SRAD(A22)                                8.
:SRAD(A33)                                8.83391257969
:SRAD(B22)                                15.5156097709
B23 | A23 | B22 | A22 |
```

## 矩阵的特征值和特征向量的定义

方阵的特征值由矩阵方程 $A \cdot x = \lambda \cdot x$ 产生。 满足该方程的 $\lambda$ 值被称为矩阵A的特征值。由I的每个值的方程得到的x的值被称为矩阵的特征向量。 有关计算特征值和特征向量的更多详细信息将在本章后面介绍。

# Function COND

函数COND确定矩阵的条件数:

```
:COND(A22)                                4.
:COND(B33)                                9.88617886179
:COND(A33)                                6.78714859438
A | X | B33 | A33 | B22 | A32
```

## 矩阵的条件数

正方形非奇异矩阵的条件数被定义为矩阵范数乘以其逆的范数的乘积，即，  
 $cond(A) = ||A|| \times ||A^{-1}||$ . We will choose as the matrix norm,  $||A||$ , the maximum of its row norm (RNRN) and column norm (CNRN), while the norm of the inverse,  $||A^{-1}||$ , will be selected as the minimum of its row norm and column norm. Thus,  $||A|| = \max(RNRN(A), CNRN(A))$ , and  $||A^{-1}|| = \min(RNRN(A^{-1}), CNRN(A^{-1}))$ .  
 $cond(A) = ||A|| \times ||A^{-1}||$ 。 我们将选择矩阵范数， $||A||$ ，其行范数（RNRN）和列范数（CNRN）的最大值，而标准反向的 $||A^{-1}||$ 将被选为其行规范的最小值列规范。 因此， $||A|| = \max(RNRN(A), CNRN(A))$  和  $||A^{-1}|| = \min(RNRN(A^{-1}), CNRN(A^{-1}))$ 。  
奇异矩阵的条件数是无穷大。 非奇异矩阵的条件数是矩阵与单数的接近程度的度量。 条件数的值越大，它就越接近奇点。（奇异矩阵是不存在逆矩阵的矩阵）。

对矩阵A33上的矩阵条件数进行以下练习。条件号是COND (A33)，行标准和A33的列标准显示在左侧。逆矩阵INV (A33) 的相应数字显示在右侧:

COND(A33)	6.78714859438	COND(INV(A33))	6.78714859437
RNRM(A33)	21.	RNRM(INV(A33))	261044176707
CNRM(A33)	20.	CNRM(INV(A33))	339357429719
HADAM LSQ MAD RANK RORM RSD		HADAM LSQ MAD RANK RORM RSD	

Since  $\text{RNRM}(\text{A33}) > \text{CNRM}(\text{A33})$ , then we take  $||\text{A33}|| = \text{RNRM}(\text{A33}) = 21$ . Also, since  $\text{CNRM}(\text{INV}(\text{A33})) < \text{RNRM}(\text{INV}(\text{A33}))$ , then we take  $||\text{INV}(\text{A33})|| = \text{CNRM}(\text{INV}(\text{A33})) = 0.261044\dots$ . Thus, the condition number is also calculated as  $\text{CNRM}(\text{A33}) * \text{CNRM}(\text{INV}(\text{A33})) = \text{COND}(\text{A33}) = 6.7871485\dots$

## Function RANK

函数RANK确定方阵的等级。请尝试以下示例:

=RANK(A22)	2
=RANK(B22)	2
A22	A22

## 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  $\mathbf{A} = [\mathbf{c}_1 \ \mathbf{c}_2 \ \dots \ \mathbf{c}_n]$ , where  $\mathbf{c}_i$  ( $i = 1, 2, \dots, n$ ) are vectors representing the columns of the matrix  $\mathbf{A}$ , then, if any of those columns, say  $\mathbf{c}_k$ ,

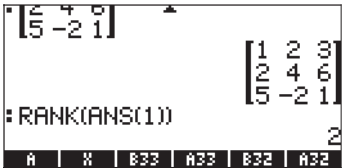
can be written as  $\mathbf{c}_k = \sum_{j \neq k, j \in \{1, 2, \dots, n\}} d_j \cdot \mathbf{c}_j$ ,

方阵的等级是矩阵包含的线性独立行或列的最大数量。假设你写了一个正方形矩阵  $A_{n \times n}$  为  $A = [c_1 \ c_2 \ \dots \ c_n]$ ，其中  $c_i$  ( $i = 1, 2, \dots, n$ ) 为向量表示矩阵  $A$  的列，然后，如果这些列中的任何一列，

值 $d_j$ 是常数，我们说 $c_k$ 线性地依赖于求和中包含的列。（注意，只要 $j \neq k$ ， $j$ 的值就包括集合 $\{1, 2, \dots, n\}$ 中的任何值，只要 $j \neq k$ 。）如果上面所示的表达式不能为任何列向量写入，那么我们就说所有列都是线性独立的。通过将矩阵写为一行行向量，可以产生类似行的线性独立性的定义。因此，如果我们发现秩 $(A) = n$ ，那么矩阵具有逆并且它是非奇异矩阵。另一方面，如果秩 $(A) < n$ ，则矩阵是单数的并且不存在逆。

where the values  $d_j$  are constant, we say that  $c_k$  is linearly dependent on the columns included in the summation. (Notice that the values of  $j$  include any value in the set  $\{1, 2, \dots, n\}$ , in any combination, as long as  $j \neq k$ .) 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  $\text{rank}(A) = n$ , then the matrix has an inverse and it is a non-singular matrix. If, on the other hand,  $\text{rank}(A) < n$ , then the matrix is singular and no inverse exist.

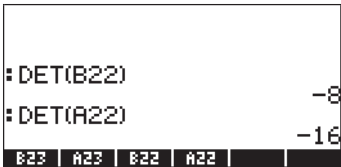
For example, try finding the rank for the matrix:



您将发现等级为2.这是因为第二行 $[2,4,6]$ 等于第一行 $[1,2,3]$ 乘以2，因此，第二行线性地依赖于第1行和 线性独立行的最大数量为2.您可以检查线性独立列的最大数量是否为3.此情况下，作为线性独立行或列的最大数量的等级变为2。

Function DET

函数DET计算方阵的行列式。例如，

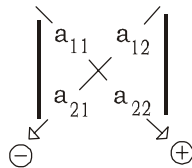


# 矩阵的行列式

2x2和/或3x3矩阵的行列式由矩阵元素的相同排列表示，但包含在垂直行之间，即，

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \quad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

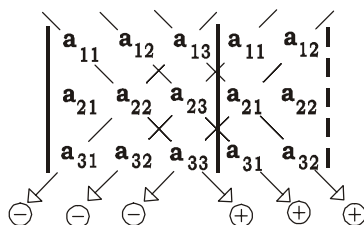
通过将其对角线中的元素相乘并添加伴有正号或负号的那些乘积来计算2x2行列式，如下图所示。



The 2x2 determinant is, therefore,

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

通过增加行列式来计算3x3行列式，该行列式包括复制行列式的前两列，并将它们放在第3列的右侧，如下图所示。该图还以与先前针对2x2行列式所做的类似方式示出了要与附加到其产品的相应符号相乘的元素。在乘法之后，将结果加在一起以获得行列式。



对于高阶方程矩阵，可以通过使用称为辅助因子的较小顺序行列式来计算。一般的想法是将 $n \times n$ 矩阵（也称为 $n \times n$ 行列式）的行列式“扩展”为辅助因子的总和，它们是 $(n-1) \times (n-1)$ 个决定因子，乘以单行或列的元素，具有交替的正负号。然后将这种“扩展”带到下一个（较低）水平，具有顺序 $(n-2) \times (n-2)$ 的辅助因子，依此类推，直到我们只剩下 $2 \times 2$ 个决定因子的长和。然后通过上面所示的方法计算 $2 \times 2$ 个决定因素。

通过辅助因子扩展计算行列式的方法是非常低效的，因为它涉及随着行列式的大小增加而增长非常快的许多操作。更有效的方法和数值应用中优选的方法是使用高斯消元法的结果。高斯消元法用于求解线性方程组。该方法的细节将在本章的后面部分介绍。

为了引用矩阵 $A$ 的行列式，我们写 $\det(A)$ 。奇异矩阵的行列式等于零。

## Function TRACE

函数TRACE计算方阵的轨迹，定义为主对角线中元素的总和

$$tr(\mathbf{A}) = \sum_{i=1}^n a_{ii}$$

Examples:

```

: TRACE(A22)          -6
: TRACE(B22)          15
B23 | A23 | B22 | A22 |

```

```

: TRACE(A33)          4
: TRACE(B33)         -7
A   | X   | B33 | A33 | B32 | A32

```

## Function TRAN

函数TRAN返回复数矩阵的实数或共轭转置的转置。TRAN等同于TRN。功能TRN的操作在第10章中介绍。

## 附加矩阵运算（矩阵OPER菜单）

矩阵OPER（OPERATIONS）可通过按键 $\leftarrow$  MATRICES 获得（系统标志117设置为CHOOSE框）:



OPERATIONS菜单包括以下功能:



功能ABS, CNRM, COND, DET, RANK, RNM, SNRM, TRACE和TRAN也可在MTH / MATRIX / NORM菜单中找到（上一节的主题）。函数SIZE在第10章中介绍。函数HADAMARD早先在矩阵乘法的上下文中介绍过。功能LSQ,





```

0:
1: « → i j « '(i+j)^2.
2: ' EVAL » »
3: 'P1'
P1 | B33 | A33 | B23 | A23 | B22 | A22

```

在这种情况下，函数LCXM的实现要求您输入：

```

2 ENTER 3 ENTER → LCXM ENTER

```

下图显示了应用函数LCXM之前和之后的RPN堆栈：

```

0:
1: « → i j « '(i+j)^2.
2: ' EVAL » »
P1 | B33 | A33 | B23 | A23 | B22

```

```

0:
1: [4. 9. 16.]
   [9. 16. 25.]
P1 | B33 | A33 | B23 | A23 | B22

```

在ALG模式下，可以通过以下方式获取此示例：

```

:LCXM(2.,3.,RCL('P1'))
[4. 9. 16.]
[9. 16. 25.]
P1 | B33 | A33 | B23 | A23 | B22

```

程序P1仍必须已创建并以RPN模式存储。

# 线性系统的解决方案

m个变量中的n个线性方程组可以写成

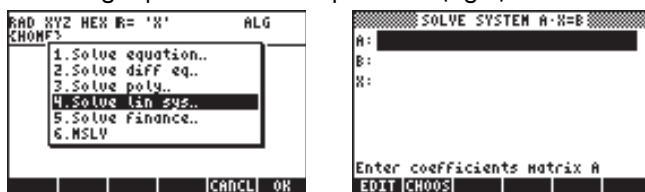
$$\begin{aligned}
 a_{11} \cdot x_1 + a_{12} \cdot x_2 + a_{13} \cdot x_3 + \dots + a_{1,m-1} \cdot x_{m-1} + a_{1,m} \cdot x_m &= b_1, \\
 a_{21} \cdot x_1 + a_{22} \cdot x_2 + a_{23} \cdot x_3 + \dots + a_{2,m-1} \cdot x_{m-1} + a_{2,m} \cdot x_m &= b_2, \\
 a_{31} \cdot x_1 + a_{32} \cdot x_2 + a_{33} \cdot x_3 + \dots + a_{3,m-1} \cdot x_{m-1} + a_{3,m} \cdot x_m &= b_3, \\
 &\vdots \\
 a_{n-1,1} \cdot x_1 + a_{n-1,2} \cdot x_2 + a_{n-1,3} \cdot x_3 + \dots + a_{n-1,m-1} \cdot x_{m-1} + a_{n-1,m} \cdot x_m &= b_{n-1}, \\
 a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + a_{n3} \cdot x_3 + \dots + a_{n,m-1} \cdot x_{m-1} + a_{n,m} \cdot x_m &= b_n.
 \end{aligned}$$

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  $\boxed{\rightarrow} \text{NUM.SLV}$ . From the numerical solver screen, shown below (left), select the option 4. *Solve lin sys..*, and press  $\boxed{\text{ENTER}}$ . The following input form will be provided (right):



To solve the linear system  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ , enter the matrix  $\mathbf{A}$ , in the format  $[[a_{11}, a_{12}, \dots], \dots [ \dots ]]$  in the A: field. Also, enter the vector  $\mathbf{b}$  in the B: field. When the X: field is highlighted, press [SOLVE]. If a solution is available, the solution vector  $\mathbf{x}$  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

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= 13, \\ x_1 - 3x_2 + 8x_3 &= -13, \\ 2x_1 - 2x_2 + 4x_3 &= -6, \end{aligned}$$

can be written as the matrix equation  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ , if

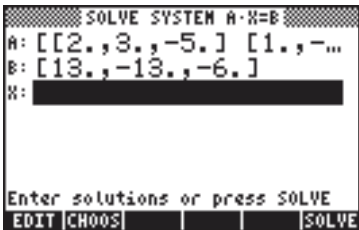
$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \\ 2 & -2 & 4 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 13 \\ -13 \\ -6 \end{bmatrix}.$$

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.

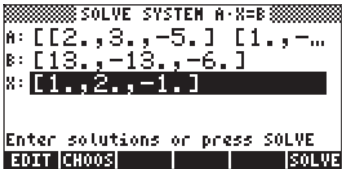
To enter matrix **A** you can activate the Matrix Writer while the A: field is selected. The following screen shows the Matrix Writer used for entering matrix **A**, as well as the input form for the numerical solver after entering matrix **A** (press **ENTER** in the Matrix Writer):



Press **▼** to select the B: field. The vector b can be entered as a row vector with a single set of brackets, i.e.,  $[13, -13, -6]$  **08** . 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:



A solution was found as shown next.



To see the solution in the stack press **ENTER** . The solution is  $\mathbf{x} = [1, 2, -1]$ .

Solutions:[1. 2. -1.]

要检查解决方案是否正确，请输入矩阵A并乘以此解决方案向量（代数模式中的示例）：

Solutions:[1. 2. -1.]

$\begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \\ 2 & -2 & 4 \end{bmatrix} \cdot \text{ANS}(1)$

[13. -13. -6.]

## Under-determined system

### 线性方程组

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= -10, \\ x_1 - 3x_2 + 8x_3 &= 85, \end{aligned}$$

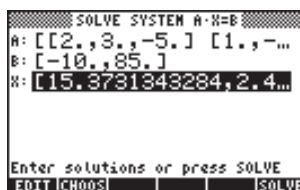
can be written as the matrix equation  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ , if

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} -10 \\ 85 \end{bmatrix}.$$

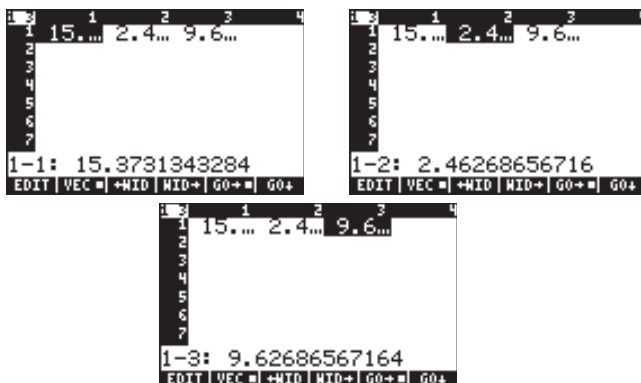
该系统具有比方程更多的未知数，因此，它不是唯一确定的。我们可以通过认识到每个线性方程表示三维平面来可视化该陈述的含义笛卡尔坐标系  $(x_1, x_2, x_3)$ 。上面所示的方程组的解决方案将是空间中两个平面的交点。然而，我们知道两个（非平行）平面的交点是直线，而不是单个点。因此，满足该系统的点不止一个。从这个意义上讲，系统并不是唯一确定的。

Let's use the numerical solver to attempt a solution to this system of equations:

NUM.SLV . Enter matrix A and vector b as illustrated in the previous example, and press when the X: field is highlighted:



To see the details of the solution vector, if needed, press the button. This will activate the Matrix Writer. Within this environment, use the right- and left-arrow keys to move about the vector:



Thus, the solution is  $\mathbf{x} = [15.373, 2.4626, 9.6268]$ .

To return to the numerical solver environment, press .

我们接下来描述的过程可用于将矩阵A和解向量X复制到堆栈中。要检查解决方案是否正确，请尝试以下操作：

- Press , 突出显示A：字段.
- Press , 将矩阵A复制到堆栈中.
- Press 返回数值求解器环境.
- Press , 以解向量X复制到堆栈中.
- Press 返回数值求解器环境.
- Press 返回堆栈.

In ALG mode, the stack will now look like this:

Solutions: [15.37313432  
[2.3, -5.]  
[1. -3. 8.]  
[15.3731343284 2.46268  
B33 | A33 | B32 | A32 | B23 | A23

Let's store the latest result in a variable X, and the matrix into variable A, as follows:

Press  $\text{STO} \rightarrow$   $\text{ALPHA}$   $X$   $\text{ENTER}$  to store the solution vector into variable X

Press  $\leftarrow$   $\leftarrow$   $\leftarrow$  to clear three levels of the stack

Press  $\text{STO} \rightarrow$   $\text{ALPHA}$   $A$   $\text{ENTER}$  to store the matrix into variable A

Now, let's verify the solution by using:  $\text{[2.3]} \times \text{[15.3731343284]} \text{ENTER}$ , which results in (press  $\nabla$  to see the vector elements): [-9.99999999992 85. ], close enough to the original vector  $\mathbf{b} = [-10 \ 85]$ .

Try also this,  $\text{[15, 10/3, 10]} \text{ENTER} \rightarrow \text{NUM} \text{ENTER}$ , i.e.,

:A\*[15 10/3 10]  
[-30. 255.]  
:→NUM(ANS(1))  
[-10. 85.]  
A | X | B33 | A33 | B32 | A32

This result indicates that  $\mathbf{x} = [15, 10/3, 10]$  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  $\mathbf{x} = [15.37... \ 2.46... \ 9.62...]$  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

$$x_1 + 3x_2 = 15,$$

$$2x_1 - 5x_2 = 5,$$







$$-x_1 + x_2 = 22,$$

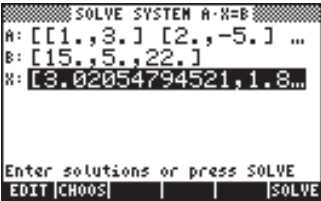
can be written as the matrix equation  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ , if


$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & -5 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 15 \\ 5 \\ 22 \end{bmatrix}.$$

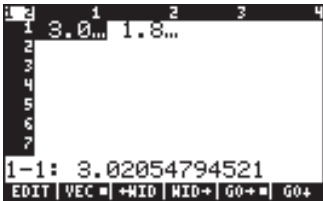
This system has more equations than unknowns (an over-determined system). The system does not have a single solution. Each of the linear equations in the system presented above represents a straight line in a two-dimensional Cartesian coordinate system ( $x_1, x_2$ ). Unless two of the three equations 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 calculator numerical solver.

Let's use the numerical solver to attempt a solution to this system of equations:

     . Enter matrix A and vector b as illustrated in the previous example, and press  when the X: field is highlighted:



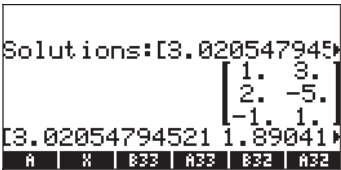
To see the details of the solution vector, if needed, press the  button. This will activate the Matrix Writer. Within this environment, use the right- and left-arrow keys to move about the vector:



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** **▣▣▣** **ENTER**, to copy matrix A onto the stack.
- Press **03** to return to the numerical solver environment.
- Press **▼▼** **▣▣▣** **ENTER**, to copy solution vector X onto the stack.
- Press **03** to return to the numerical solver environment.
- Press **ENTER** to return to the stack.

In ALG mode, the stack will now look like this:



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: **▣▣▣** **×** **▣▣▣** **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.

## 最小二乘解 (函数LSQ)

LSQ函数根据以下标准返回线性系统 $Ax = b$ 的最小范数最小二乘解:



- 如果A是方阵并且A是非奇异的（即，它存在逆矩阵，或者其行列式不为零），则LSQ将精确解返回到线性系统。
- 如果A具有小于满行级别（欠定方程组），则LSQ返回具有无穷多个解的最小欧几里德长度的解。
- 如果A具有小于满列的列（超定方程组），则LSQ返回具有最小残值 $e = \|A \cdot x - b\|$ 的“解”。方程组可能没有解，因此，返回的值不是系统的真正解，只是具有最小残差的系统。

函数LSQ按顺序作为输入向量b和矩阵A。函数LSQ可以在函数目录(  CAT ) 中找到。接下来，我们使用函数LSQ重复前面使用数值解算器找到的解：

## Square system

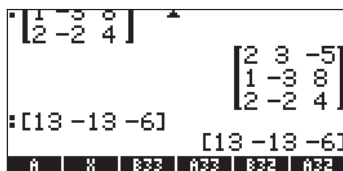
Consider the system

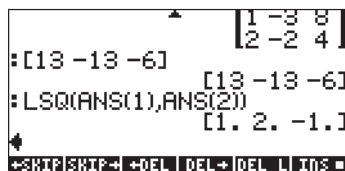
$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= 13, \\ x_1 - 3x_2 + 8x_3 &= -13, \\ 2x_1 - 2x_2 + 4x_3 &= -6, \end{aligned}$$

with

$$A = \begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \\ 2 & -2 & 4 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \text{and} \quad b = \begin{bmatrix} 13 \\ -13 \\ -6 \end{bmatrix}.$$

The solution using LSQ is shown next:





## Under-determined system

Consider the system

$$2x_1 + 3x_2 - 5x_3 = -10,$$

$$x_1 - 3x_2 + 8x_3 = 85,$$

with

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} -10 \\ 85 \end{bmatrix}.$$

The solution using LSQ is shown next:

## Over-determined system

Consider the system

$$x_1 + 3x_2 = 15,$$

$$2x_1 - 5x_2 = 5,$$

$$-x_1 + x_2 = 22,$$

with

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & -5 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 15 \\ 5 \\ 22 \end{bmatrix}.$$

The solution using LSQ is shown next:

将这三种解决方案与使用数值解算器计算的解决方案进行比较。

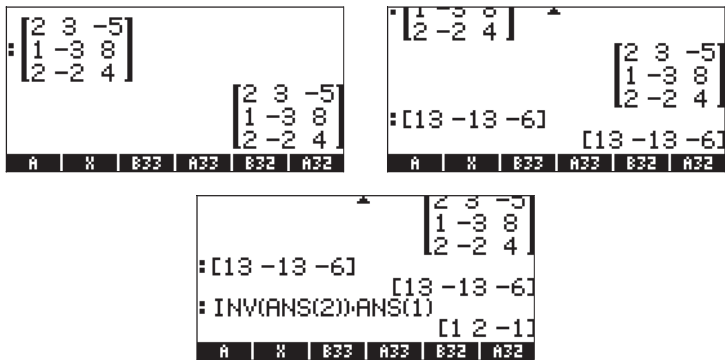
### 用逆矩阵求解

The solution to the system  $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ , where  $\mathbf{A}$  is a square matrix is  $\mathbf{x} = \mathbf{A}^{-1}\cdot\mathbf{b}$ . This results from multiplying the first equation by  $\mathbf{A}^{-1}$ , i.e.,  $\mathbf{A}^{-1}\cdot\mathbf{A}\cdot\mathbf{x} = \mathbf{A}^{-1}\cdot\mathbf{b}$ . By definition,  $\mathbf{A}^{-1}\cdot\mathbf{A} = \mathbf{I}$ , thus we write  $\mathbf{I}\cdot\mathbf{x} = \mathbf{A}^{-1}\cdot\mathbf{b}$ . Also,  $\mathbf{I}\cdot\mathbf{x} = \mathbf{x}$ , thus, we have,  $\mathbf{x} = \mathbf{A}^{-1}\cdot\mathbf{b}$ .

For the example used earlier, namely,

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= 13, \\ x_1 - 3x_2 + 8x_3 &= -13, \\ 2x_1 - 2x_2 + 4x_3 &= -6, \end{aligned}$$

we can find the solution in the calculator as follows:



这与之前发现的结果相同。

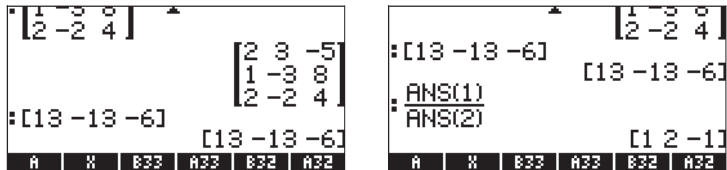
### 矩阵“划分”的解决方案

While the operation of division is not defined for matrices, we can use the calculator's  $\div$  key to “divide” vector  $\mathbf{b}$  by matrix  $\mathbf{A}$  to solve for  $\mathbf{x}$  in the matrix equation  $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ . This is an arbitrary extension of the algebraic division operation to matrices, i.e., from  $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ , we dare to write  $\mathbf{x} = \mathbf{b}/\mathbf{A}$  (Mathematicians would cringe if they see this!) This, of course is interpreted as  $(1/\mathbf{A})\cdot\mathbf{b} = \mathbf{A}^{-1}\cdot\mathbf{b}$ , which is the same as using the inverse of  $\mathbf{A}$  as in the previous section.

下面针对该情况说明了将B“除以”A的情况的过程

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= 13, \\ x_1 - 3x_2 + 8x_3 &= -13, \\ 2x_1 - 2x_2 + 4x_3 &= -6, \end{aligned}$$

该过程显示在以下屏幕截图中：



与上面使用逆矩阵找到的解决方案相同。

## 用相同的系数矩阵求解多组方程

假设您要解决以下三组方程：

$$\begin{aligned} X + 2Y + 3Z &= 14, & 2X + 4Y + 6Z &= 9, & 2X + 4Y + 6Z &= -2, \\ 3X - 2Y + Z &= 2, & 3X - 2Y + Z &= -5, & 3X - 2Y + Z &= 2, \\ 4X + 2Y - Z &= 5, & 4X + 2Y - Z &= 19, & 4X + 2Y - Z &= 12. \end{aligned}$$

我们可以将三个方程组写成单个矩阵方程： **$A \cdot X = B$** ，其中

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & -2 & 1 \\ 4 & 2 & -1 \end{bmatrix}, \quad X = \begin{bmatrix} X_{(1)} & X_{(2)} & X_{(3)} \\ Y_{(1)} & Y_{(2)} & Y_{(3)} \\ Z_{(1)} & Z_{(2)} & Z_{(3)} \end{bmatrix},$$

$$B = \begin{bmatrix} 14 & 9 & -2 \\ 2 & -5 & 2 \\ 5 & 19 & 12 \end{bmatrix}.$$

变量名称X, Y和Z中的子索引确定它们所指的方程系统。要解决此扩展系统，我们在RPN模式下使用以下过程：

```
[[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:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 5 & 1 \\ 3 & -1 & -2 \end{bmatrix}.$$

## 高斯和高斯 - 乔丹消除

高斯消元是一种过程，通过该过程，属于 $n$ 个未知数的 $n$ 个线性方程组的系数的方阵通过一系列行操作被简化为上三角矩阵（梯形形式）。此过程称为前向消除。将系数矩阵简化为上三角形形式允许在称为后向替换的过程中一次仅利用一个方程来求解所有 $n$ 个未知数。

### 使用方程的高斯消元的示例

为了说明高斯消元法，我们将在3个未知数中使用以下3个方程组：

$$2X + 4Y + 6Z = 14,$$

$$3X - 2Y + Z = -3,$$

$$4X + 2Y - Z = -4.$$

我们可以将这些方程分别存储在变量E1，E2和E3中的计算器中，如下所示。出于备份目的，还创建了包含三个方程的列表并将其存储到变量EQS中。这样，如果出现错误，方程式仍然可供用户使用。

```
: 2X+4Y+6Z=14►E1
      2X+4Y+6Z=14
: 3X-2Y+Z=-3►E2
      3X-(2Y-Z)=-3
: 4X+2Y-Z=-4►E3
      4X+2Y-Z=-4
EQS | EQS | EQS | EQS | EQS |
```

```
: E1
      2X+4Y+6Z=14
: E2
      3X-(2Y-Z)=-3
: E3
      4X+2Y-Z=-4
EQS | EQS | EQS | EQS | EQS |
```

为了开始前向消除过程，我们将第一个等式（E1）除以2，并将其存储在E1中，并再次显示三个等式以产生：

: E1	→ E1				
					X+2Y+3Z=7
: E2					
					3X-(2Y-Z)=-8
: E3					
					4X+2Y-Z=-4
EQS   E3   E2   E1					

接下来，我们用（方程式2-3-3方程1，即E1-3×E2）代替第二方程式E2，用方程式3-4×方程式代替第三方程式得到：

: E2-3·E1	→ E2				
					-(8Y+8Z-24)
: E3-4·E1	→ E3				
					-(6Y+13Z-32)
EQS   E3   E2   E1					

: E1					X+2Y+3Z=7
: E2					-(8Y+8Z-24)
: E3					-(6Y+13Z-32)
EQS   E3   E2   E1					

接下来，将第二个等式除以-8，得到：

: E2	→ E2				
					Y+Z=3
EQS   E3   E2   E1					

: E2	→ E2				
					Y+Z=3
EQS   E3   E2   E1					

接下来，用（等式3 + 6×等式2，即E2 + 6×E3）代替第三个等式E3，得到：

: E3+6·E2	→ E3				
					-(7Z-14)
EQS   E3   E2   E1					

: E1					X+2Y+3Z=7
: E2					Y+Z=3
: E3					-(7Z-14)
EQS   E3   E2   E1					

请注意，当我们执行方程的线性组合时，计算器将结果修改为等号左侧的表达式，即

表达式= 0.因此，最后一组方程被解释为以下等价方程组：

$$\begin{aligned} X + 2Y + 3Z &= 7, \\ Y + Z &= 3, \\ -7Z &= -14. \end{aligned}$$

高斯消元中的后向替换过程包括从最后一个方程开始并向上工作，找到未知数的值。因此，我们首先解决Z：

X+2Y+3Z=7

: E2

Y+Z=3

: E3

-(7Z-14)

: SOLVE(E3,'Z')

Z=2

Eqs

E3

E2

E1

Y+Z=3

: E3

-(7Z-14)

: SOLVE(E3,'Z')

Z=2

: SUBST(E2,ANS(1))>>E2

Y+2=3

Eqs

E3

E2

E1

接下来，我们将Z = 2代入等式2（E2），并为Y求解E2：

-(7Z-14)

: SOLVE(E3,'Z')

Z=2

: SUBST(E2,ANS(1))>>E2

Y+2=3

: SOLVE(E2,'Y')

Y=1

Eqs

E3

E2

E1

接下来，我们将Z = 2和Y = 1替换为E1，并解决E1的X：

Y=1

: SUBST(E1,Y=1)

X+2\*1+3\*Z=7

: SUBST(ANS(1),Z=2)

X+2\*1+3\*Z=7

: ANS(1)>>E1

X+2\*1+3\*Z=7

Eqs

E3

E2

E1

CASDI

X+2\*1+3\*Z=7

: SUBST(ANS(1),Z=2)

X+2\*1+3\*Z=7

: ANS(1)>>E1

X+2\*1+3\*Z=7

: SOLVE(ANS(1),'X')

X=-1

Eqs

E3

E2

E1

CASDI

因此，解决方案是X = -1，Y = 1，Z = 2。

使用矩阵的高斯消元的示例

上面例子中使用的方程组可以写成矩阵方程A x= b，如果我们使用：

$$\mathbf{A} = \begin{pmatrix} 2 & 4 & 6 \\ 3 & -2 & 1 \\ 4 & 2 & -1 \end{pmatrix}, \quad \mathbf{x} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 14 \\ -3 \\ -4 \end{bmatrix}.$$

为了使用高斯消元法获得系统矩阵方程的解，我们首先创建所谓的对应于 $\mathbf{A}$ 的增广矩阵，即，

$$\mathbf{A}_{aug} = \left( \begin{array}{ccc|c} 2 & 4 & 6 & 14 \\ 3 & -2 & 1 & -3 \\ 4 & 2 & -1 & -4 \end{array} \right)$$

矩阵matrix  $\mathbf{A}_{aug}$  与原始矩阵 $\mathbf{A}$ 相同，其中对应于向量 $\mathbf{b}$ 的元素的新行被添加（即，增强）到 $\mathbf{A}$ 的最右列的右侧。

一旦将增强矩阵放在一起，我们就可以继续对其执行行操作，这将把原始 $\mathbf{A}$ 矩阵减少为上三角矩阵。在本练习中，我们将使用RPN模式(MODE  $\frac{+}{-}$   $\frac{1}{x}$   $\frac{1}{y}$ )，系统标志117设置为SOFT菜单。在计算器中，使用以下按键。首先，输入增强矩阵，并在堆栈中制作相同的额外副本（此步骤不是必需的，除非作为保险，如果您在前向消除中出错，则保存增加的矩阵的额外副本 我们即将进行的程序。）：

[[2,4,6,14],[3,-2,1,-3],[4,2,-1,-4]] ENTER ENTER

在变量AAUG中保存增广矩阵:  $\frac{1}{x}$  ALPHA ALPHA A A U G ALPHA STO▶

使用堆栈中增强矩阵的副本，按  $\frac{1}{x}$  MTH  $\frac{1}{x}$   $\frac{1}{y}$  激活ROW操作菜单。

接下来，在增强矩阵上执行以下行操作。






乘以行1 by  $\frac{1}{2}$ : 2  $\frac{1}{x}$   $\frac{1}{y}$   $\frac{1}{x}$






将第1行乘以-3将其添加到第2行，替换它: 3  $\frac{+}{-}$  SPC  $\frac{1}{x}$  SPC 2

$\frac{1}{x}$

将第1行乘以-4将其添加到第3行，替换它: 4  $\frac{+}{-}$  SPC  $\frac{1}{x}$  SPC 3  $\frac{1}{x}$



将第2行乘以-1/8:     

将第2行乘以6将其添加到第3行，替换它:     

如果您手动执行这些操作，则应编写以下内容：

$$\mathbf{A}_{aug} = \left( \begin{array}{ccc|c} 2 & 4 & 6 & 14 \\ 3 & -2 & 1 & -3 \\ 4 & 2 & -1 & -4 \end{array} \right) \cong \left( \begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 3 & -2 & 1 & -3 \\ 4 & 2 & -1 & -4 \end{array} \right)$$
$$\mathbf{A}_{aug} \cong \left( \begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & -8 & -8 & -24 \\ 0 & -6 & -13 & -32 \end{array} \right) \cong \left( \begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 3 \\ 0 & -6 & -13 & -32 \end{array} \right)$$
$$\mathbf{A}_{aug} \cong \left( \begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & -7 & -14 \end{array} \right)$$

符号 $\cong$ （“相当于”）表示后面的内容等同于前一个矩阵，其中涉及一些行（或列）操作。






得到的矩阵是上三角形，等价于方程组







$$\begin{aligned} X + 2Y + 3Z &= 7, \\ Y + Z &= 3, \\ -7Z &= -14, \end{aligned}$$

现在可以解决，一次一个方程，通过后向替换，如







### 使用矩阵消除高斯 - 乔丹







高斯 - 约旦消除包括继续由前向消除过程产生的上三角矩阵中的行操作，直到单位矩阵代替原始 $\mathbf{A}$ 矩阵。例如，对于我们刚刚介绍的情况，我们可以继续执行以下行操作：

将行3乘以-1/7:     

将第3行乘以-1, 将其添加到第2行, 替换它:      

将第3行乘以-3, 将其添加到第1行, 替换它:

将第2行乘以-2, 将其添加到第1行, 替换它:      

手动编写此过程将导致以下步骤:

$$\mathbf{A}_{aug} = \left( \begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & -7 & -14 \end{array} \right) \cong \left( \begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 1 & 3 \\ 0 & 0 & 1 & 2 \end{array} \right) \cong \left( \begin{array}{ccc|c} 1 & 2 & 3 & 7 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array} \right)$$

$$\mathbf{A}_{aug} \cong \left( \begin{array}{ccc|c} 1 & 2 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array} \right) \cong \left( \begin{array}{ccc|c} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array} \right).$$

## 旋转

如果仔细查看上面示例中的行操作, 您会注意到许多操作将行与主对角线中的相应元素分开。此元素称为枢轴元素, 或简称为枢轴。在许多情况下, 枢轴元件可能变为零, 在这种情况下, 我们不能通过其枢轴划分行。此外, 为了改进使用高斯或高斯 - 约旦消除的方程组的数值解, 建议枢轴是给定列中具有最大绝对值的元素。在这种情况下, 我们在执行行操作之前交换行。这种行交换称为部分旋转。为了遵循这一建议, 通常需要在执行高斯或高斯 - 约旦消除时交换增广矩阵中的行。

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 full pivoting.

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 permutation matrix  $\mathbf{P} = \mathbf{I}_{n \times n}$ , at the beginning of the procedure. Any row or column exchange required in the augmented matrix  $\mathbf{A}_{\text{aug}}$  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  $\mathbf{x}$  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:






$$\begin{aligned} X + 2Y + 3Z &= 2, \\ 2X + \quad \quad 3Z &= -1, \\ 8X + 16Y - Z &= 41. \end{aligned}$$

The augmented matrix and the permutation matrix are as follows:







$$\mathbf{A}_{\text{aug}} = \begin{bmatrix} 1 & 2 & 3 & 2 \\ 2 & 0 & 3 & -1 \\ 8 & 16 & -1 & 41 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$


Store the augmented matrix in variable AAUG, then press  $\boxed{\rightarrow} \boxed{\text{AAUG}}$  to get a copy in the stack. We want to keep the CSWP (Column Swap) command readily available, for which we use:  $\boxed{\rightarrow} \boxed{\text{CAT}} \boxed{\text{ALPHA}} \boxed{\text{ALPHA}} \boxed{\text{C}} \boxed{\text{S}} \boxed{\text{ALPHA}}$  (find CSWP),  $\boxed{\text{OK}}$ . You'll get an error message, press  $\boxed{\text{ON}}$ , and ignore the message. Next, get the ROW menu available by pressing:  $\boxed{\leftarrow} \boxed{\text{MATRICES}} \boxed{\text{DOWN}} \boxed{\text{DOWN}}$ .

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:     . The augmented matrix and the permutation matrix now are:







$$\left[ \begin{array}{cccc} 8 & 16 & -1 & 41 \\ 2 & 0 & 3 & -1 \\ 1 & 2 & 3 & 2 \end{array} \right] \quad \left[ \begin{array}{ccc} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{array} \right]$$

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:      .

 The augmented matrix and the permutation matrix now are:









$$\left[ \begin{array}{cccc} 16 & 8 & -1 & 41 \\ 0 & 2 & 3 & -1 \\ 2 & 1 & 3 & 2 \end{array} \right] \quad \left[ \begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array} \right]$$

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:


     . The permutation matrix does not change, but the augmented matrix is now:

$$\left[ \begin{array}{cccc} 1 & 1/2 & -1/16 & 41/16 \\ 0 & 2 & 3 & -1 \\ 2 & 1 & 3 & 2 \end{array} \right] \quad \left[ \begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array} \right]$$

The next step is to eliminate the 2 from position (3,2) by using:


       .

$$\left[ \begin{array}{cccc} 1 & 1/2 & -1/16 & 41/16 \\ 0 & 2 & 3 & -1 \\ 0 & 0 & 25/8 & -25/8 \end{array} \right] \quad \left[ \begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array} \right]$$

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** 

$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 3 & 2 & -1 \\ 0 & 25/8 & 0 & -25/8 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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** 


$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 25/8 & 0 & -25/8 \\ 0 & 3 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Now, we are ready to divide row 2 by the pivot 25/8, by using


**1** **8** **÷** **2** **5** **▶** **SPC** **2** **NXT** 

$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 1 & 0 & -1 \\ 0 & 3 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Next, we eliminate the 3 from position (3,2) by using:


**3** **+/-** **SPC** **2** **SPC** **3** 

$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 2 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

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 1/2 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** **/x** **3** 


$$\begin{bmatrix} 1 & -1/16 & 1/2 & 41/16 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Next, we proceed to eliminate the 1/2 in position (1,3) by using:

2  $\frac{1}{x}$  +/- SPC 3 SPC / 

$$\begin{bmatrix} 1 & -1/16 & 0 & 33/16 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Finally, we eliminate the  $-1/16$  from position (1,2) by using:

/ 6  $\frac{1}{x}$  SPC 2 SPC / 

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

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  $\mathbf{P}$ . We identify the unknown vector  $\mathbf{x}$ , the modified independent vector  $\mathbf{b}'$  and the permutation matrix  $\mathbf{P}$  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  $\mathbf{P} \cdot \mathbf{x} = \mathbf{b}'$ , 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:



$$\mathbf{A}_{aug(I)} = \left[ \begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 3 & -2 & 1 & 0 & 1 & 0 \\ 4 & 2 & -1 & 0 & 0 & 1 \end{array} \right].$$

To see the intermediate steps in calculating and inverse, just enter the matrix **A** from above, and press  $\frac{1}{x}$ , while keeping the step-by-step option active in the calculator's CAS. Use the following:

$\left[ \left[ 1, 2, 3 \right], \left[ 3, -2, 1 \right], \left[ 4, 2, -1 \right] \right] \text{ (ENTER) } \frac{1}{x}$

After going through the different steps, the solution returned is:

$$\begin{bmatrix} 0 & \frac{1}{7} & \frac{1}{7} \\ \frac{1}{8} & -\frac{13}{56} & \frac{1}{7} \\ \frac{1}{4} & \frac{3}{28} & -\frac{1}{7} \end{bmatrix}$$

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  $\det(\mathbf{A}) = 56$ .

We could write,  $\mathbf{A}^{-1} = \mathbf{C} / \det(\mathbf{A})$ , where **C** is the matrix


$$\mathbf{C} = \begin{bmatrix} 0 & 8 & 8 \\ 7 & -13 & 8 \\ 14 & 6 & -8 \end{bmatrix}.$$

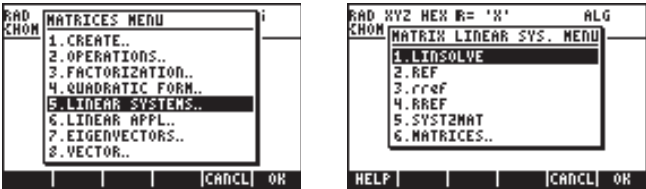


The result  $(\mathbf{A}^{-1})_{n \times n} = \mathbf{C}_{n \times n} / \det(\mathbf{A}_{n \times n})$ , is a general result that applies to any non-singular matrix  $\mathbf{A}$ . A general form for the elements of  $\mathbf{C}$  can be written based on the Gauss-Jordan algorithm.

Based on the equation  $\mathbf{A}^{-1} = \mathbf{C} / \det(\mathbf{A})$ , sketched above, the inverse matrix,  $\mathbf{A}^{-1}$ , is not defined if  $\det(\mathbf{A}) = 0$ . Thus, the condition  $\det(\mathbf{A}) = 0$  defines also a singular matrix.


### Solution to linear systems using calculator functions

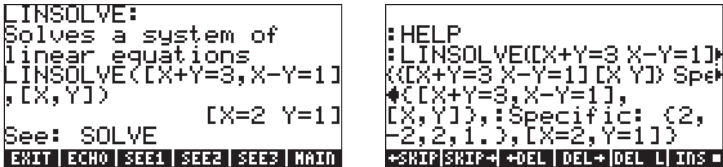
The simplest way to solve a system of linear equations,  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ , in the calculator is to enter  $\mathbf{b}$ , enter  $\mathbf{A}$ , and then use the division function  $/$ . If the system of linear equations is over-determined or under-determined, a “solution” can 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):



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,  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ , where

$$\mathbf{A} = [[1, -2, 1], [2, 1, -2], [5, -2, 1]],$$

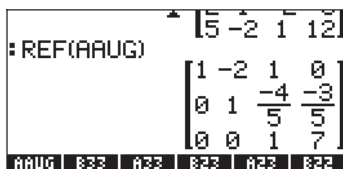
and

$$\mathbf{b} = [[0], [-3], [12]].$$

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:



$$\begin{array}{c} [5 \ -2 \ 1 \ 12] \\ \text{REF(AAUG)} \\ \left[ \begin{array}{ccc|c} 1 & -2 & 1 & 0 \\ 0 & 1 & -4 & -3 \\ 0 & 0 & 1 & 7 \end{array} \right] \\ \text{AAUG} \quad \text{B33} \quad \text{A33} \quad \text{B23} \quad \text{A23} \quad \text{B22} \end{array}$$

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 result 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:



The result is final augmented matrix resulting from a Gauss-Jordan elimination without pivoting.

A row-reduced echelon form 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:



The second screen above is obtained by activating the line editor (press  $\nabla$ ). 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])
+SKIP+SKIP+ +DEL DEL+ DEL L INS =

```

```

:HELP
:SYST2MAT([X+Y X-Y=2],[X
:SYST2MAT([X+Y,X-Y=2],
[X,Y])
+SKIP+SKIP+ +DEL DEL+ DEL L INS =

```

The result is the augmented matrix corresponding to the system of equations:

$$X+Y = 0$$

$$X-Y = 2$$

## 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,  $\mathbf{e}$  is a vector of residuals of Function for the vector  $\mathbf{x} = \mathbf{x}(0)$ .

To use Function RSD you need the terms  $\mathbf{b}$ ,  $\mathbf{A}$ , and  $\mathbf{x}(0)$ , as arguments. The vector returned is  $\mathbf{e} = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}(0)$ . For example, using  $\mathbf{A} = \begin{bmatrix} 2 & -1 \\ 0 & 2 \end{bmatrix}$ ,  $\mathbf{x}(0) = [1.8, 2.7]$ , and  $\mathbf{b} = [1, 6]$ , we can find the vector of residuals as follows:

```

RSD([1,6],[[2,-1],[0,
2]], [1.8,2.7])
+SKIP+SKIP+ +DEL DEL+ DEL L INS =

```

```

:RSD([1 6],[[2 -1],
[0 2]], [1.8 2.7])
+SKIP+SKIP+ +DEL DEL+ DEL L INS =

```

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  $\mathbf{x}(0)$ , 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  $\mathbf{A}$ , we can write the eigenvalue equation  $\mathbf{A} \cdot \mathbf{x} = \lambda \cdot \mathbf{x}$ , where the values of  $\lambda$  that satisfy the equation are known as the eigenvalues of matrix  $\mathbf{A}$ . For each value of  $\lambda$ , we can find, from the same equation, values of  $\mathbf{x}$  that satisfy the eigenvalue equation. These values of  $\mathbf{x}$  are known as the eigenvectors of matrix  $\mathbf{A}$ . The eigenvalues equation can be written also as  $(\mathbf{A} - \lambda \cdot \mathbf{I})\mathbf{x} = 0$ .

This equation will have a non-trivial solution only if the matrix  $(\mathbf{A} - \lambda \cdot \mathbf{I})$  is singular, i.e., if  $\det(\mathbf{A} - \lambda \cdot \mathbf{I}) = 0$ .

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  $\mathbf{A}$ . 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]]



Using the variable  $\lambda$  to represent eigenvalues, this characteristic polynomial is to be interpreted as  $\lambda^3 - 2\lambda^2 - 22\lambda + 21 = 0$ .

### Function EGV

Function EGV (EiGenValues) produces the eigenvalues of a square matrix. For example, the eigenvalues of the matrix shown below are calculated in ALG mode using function EGV:



The eigenvalues  $\lambda = [-\sqrt{10}, \sqrt{10}]$ .

**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 EGV. 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:



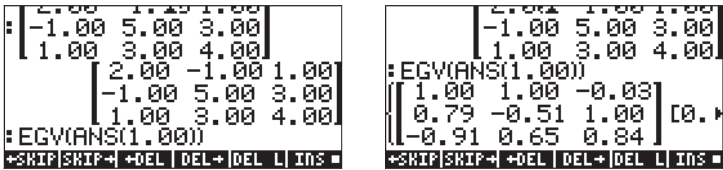
Change mode to Approx and repeat the entry, to get the following eigenvalues:  $[(1.38, 2.22), (1.38, -2.22), (-1.76, 0)]$ .

### 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:



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:



In summary,

$$\begin{aligned} \lambda_1 &= 0.29, \mathbf{x}_1 = [1.00, 0.79, -0.91]^T, \\ \lambda_2 &= 3.16, \mathbf{x}_2 = [1.00, -0.51, 0.65]^T, \\ \lambda_3 &= 7.54, \mathbf{x}_3 = [-0.03, 1.00, 0.84]^T. \end{aligned}$$

**Note:** A symmetric matrix produces all real eigenvalues, and its eigenvectors are mutually perpendicular. For the example just worked out, you can check that  $\mathbf{x}_1 \cdot \mathbf{x}_2 = 0$ ,  $\mathbf{x}_1 \cdot \mathbf{x}_3 = 0$ , and  $\mathbf{x}_2 \cdot \mathbf{x}_3 = 0$ .

## 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:



## 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 generates 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  $(\mathbf{x} \cdot \mathbf{I} - \mathbf{A}) \cdot \mathbf{p}(\mathbf{x}) = \mathbf{m}(\mathbf{x}) \cdot \mathbf{I}$ ,
- the characteristic polynomial of the matrix (stack level 1)



Notice that the equation  $(\mathbf{x} - \mathbf{I}\mathbf{A}) \cdot \mathbf{p}(\mathbf{x}) = \mathbf{m}(\mathbf{x}) \cdot \mathbf{I}$  is similar, in form, to the eigenvalue equation  $\mathbf{A} \cdot \mathbf{x} = \lambda \cdot \mathbf{x}$ .

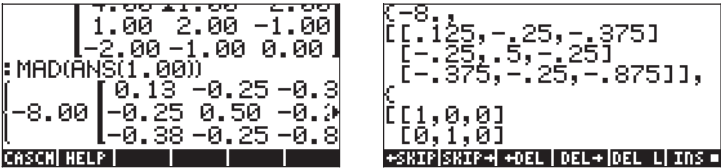
As an example, in RPN mode, try:

```
[ [4,1,-2] [1,2,-1] [-2,-1,0] ] MAD
```

The result is:

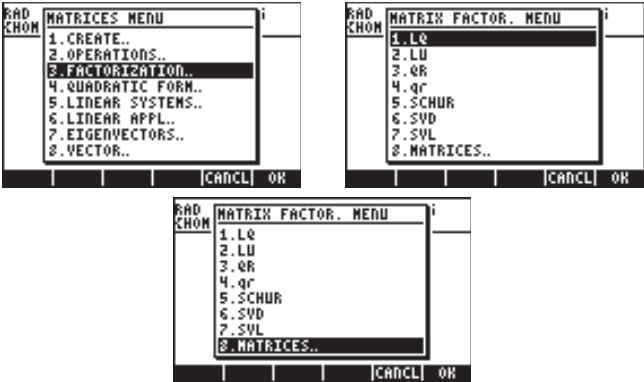
```
4: -8.  
3: [[ 0.13 -0.25 -0.38] [-0.25 0.50 -0.25] [-0.38 -0.25 -0.88]]  
2: {[[[1 0 0][0 1 0][0 0 1]] [[ -2 1 -2][1 -4 -1][2 -1 -6]] [[-1 2 3][2 -4 2][3 2 7]]}  
1: 'X^3+6*x^2+2*X+8'
```

The same exercise, in ALG mode, will look as follows:



# 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.

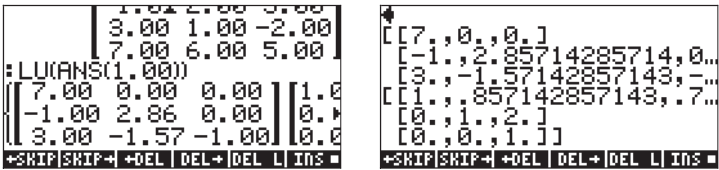
## Function LU

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  $\mathbf{P} \cdot \mathbf{A} = \mathbf{L} \cdot \mathbf{U}$ . 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:



## 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  $\mathbf{U} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_n]$  where the  $\mathbf{v}_i$ ,  $i = 1, 2, \dots, n$ , are column vectors, and if  $\mathbf{v}_i \cdot \mathbf{v}_j = \delta_{ij}$ , where  $\delta_{ij}$  is the Kronecker's delta function, then  $\mathbf{U}$  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  $\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \cdot \mathbf{S}_{m \times n} \cdot \mathbf{V}_{n \times n}^T$  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  $i = 1, 2, \dots, n-1$ . The columns  $[\mathbf{u}_i]$  of **U** and  $[\mathbf{v}_i]$  of **V** are the corresponding singular vectors.

## 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  $n \times m$  matrix is the equivalent of a triangular matrix out of an  $n \times n$  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  $\mathbf{A}$ ,  $\mathbf{P}$ ,  $\mathbf{Q}$  and  $\mathbf{R}$  are related by  $\mathbf{A} \cdot \mathbf{P} = \mathbf{Q} \cdot \mathbf{R}$ . For example,  $\begin{bmatrix} 1 & -2 & 1 \\ 2 & 1 & -2 \\ 5 & 2 & 1 \end{bmatrix}$  QR produces

3:  $\begin{bmatrix} -0.18 & 0.39 & 0.90 \\ -0.37 & -0.88 & 0.30 \\ -0.91 & 0.28 & -0.30 \end{bmatrix}$   
 2:  $\begin{bmatrix} -5.48 & -0.37 & 1.83 \\ 0 & 2.42 & -2.20 \\ 0 & 0 & -0.90 \end{bmatrix}$   
 1:  $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

**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 quadratic form from a square matrix  $\mathbf{A}$  is a polynomial expression originated from  $\mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}^T$ . For example, if we use  $\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 5 & 4 & 2 \\ 3 & 5 & -1 \end{bmatrix}$ , and  $\mathbf{x} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ , the corresponding quadratic form is calculated as

$$\begin{aligned} \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}^T &= \begin{bmatrix} X & Y & Z \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 & -1 \\ 5 & 4 & 2 \\ 3 & 5 & -1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\ &= \begin{bmatrix} X & Y & Z \end{bmatrix} \cdot \begin{bmatrix} 2X + Y - Z \\ 5X + 4Y + 2Z \\ 3X + 5Y - Z \end{bmatrix} \end{aligned}$$

Finally,  $\mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}^T = 2X^2 + 4Y^2 - Z^2 + 6XY + 2XZ + 7ZY$

## The QUADF menu

The 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:  $2X^2 + (6Y + 2Z)X + 4Y^2 + 7Z^2$

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  $\mathbf{A}$  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:  $\begin{bmatrix} 1 & 2 & -8 \\ 2 & 1 & 0 \\ -8 & 0 & -1 \end{bmatrix}$

1: ['X' 'Y' 'Z']

## Diagonal representation of a quadratic form

Given a symmetric square matrix  $\mathbf{A}$ , it is possible to “diagonalize” the matrix  $\mathbf{A}$  by finding an orthogonal matrix  $\mathbf{P}$  such that  $\mathbf{P}^T \cdot \mathbf{A} \cdot \mathbf{P} = \mathbf{D}$ , where  $\mathbf{D}$  is a diagonal matrix. If  $Q = \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}^T$  is a quadratic form based on  $\mathbf{A}$ , it is possible to write the quadratic form  $Q$  so that it only contains square terms from a variable  $\mathbf{y}$ ,

such that  $\mathbf{x} = \mathbf{P} \cdot \mathbf{y}$ , by using  $\mathbf{Q} = \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}^T = (\mathbf{P} \cdot \mathbf{y}) \cdot \mathbf{A} \cdot (\mathbf{P} \cdot \mathbf{y})^T = \mathbf{y} \cdot (\mathbf{P}^T \cdot \mathbf{A} \cdot \mathbf{P}) \cdot \mathbf{y}^T = \mathbf{y} \cdot \mathbf{D} \cdot \mathbf{y}^T$ .

### Function SYLVESTER

函数SYLVESTER将对称方阵**A**作为参数，并返回包含对角矩阵**D**的对角项和矩阵**P**的向量，以便  $\mathbf{P}^T \cdot \mathbf{A} \cdot \mathbf{P} = \mathbf{D}$ 。例如：

`[[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  $\mathbf{Q} = \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}^T$  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 .



有关此菜单中列出的功能的信息，请参阅下面的计算器自己的帮助工具。这些数字显示了帮助设施条目和附带的示例。

## 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
EXIT ECHO SEE1 SEE2 SEE3 MAIN

```

```

:HELP
: IMAGE([[1 2 3]
         [4 5 6]])
      [[1 0] [0 1]]
CASCM HELP

```

## Function ISOM

```

ISOM:
Finds elements of a
2-d or 3-d linear
isometry
ISOM([[0,-1],[1,0]])
      {π/2 1}
See: MKISOM
EXIT ECHO SEE1 SEE2 SEE3 MAIN

```

```

:HELP
: ISOM([[0 -1]
        [1 0]])
      {π/2 1}
CASCM HELP

```

## Function KER

```
KER:
Kernel of a linear ap-
plication of matrix M
KER([[1,2,3],[4,5,6]])
      <[-1 2 -1]>
See: IMAGE
EXIT ECHO SEE1 SEE2 SEE3 MAIN
```

```
:HELP
:KER([[1 2 3]
      [4 5 6]
      <[-1 2 -1]>
CASCM HELP
```

## Function MKISOM

```
MKISOM:
Make an isometry given
its elements
MKISOM( $\pi$ ,1)
      [[-1,0],[0,-1]]
See: ISOM
EXIT ECHO SEE1 SEE2 SEE3 MAIN
```

```
:HELP
:MKISOM( $\pi$ ,1)
      [-1 0]
      [ 0 -1]
CASCM HELP
```