

Chapter 11

Matrix Operations and Linear Algebra

In Chapter 10 we introduced the concept of a matrix and presented a number of functions for entering, creating, or manipulating matrices. In this Chapter we present examples of matrix operations and applications to problems of linear algebra.

Operations with matrices

Matrices, like other mathematical objects, can be added and subtracted. They can be multiplied by a scalar, or among themselves. They can also be raised to a real power. An important operation for linear algebra applications is the inverse of a matrix. Details of these operations are presented next.



To illustrate the operations we will create a number of matrices that we will store in variables. The generic name of the matrices will be A_{ij} and B_{ij} , where i represents the number of rows and j the number of columns of the matrices. The matrices to be used are generated by using function RANM (random matrices). If you try this exercise in your calculator you will get different matrices than the ones listed herein, unless you store them into your calculator exactly as shown below. Here are the matrices A22, B22, A23, B23, A32, B32, A33 and B33 created in ALG mode:

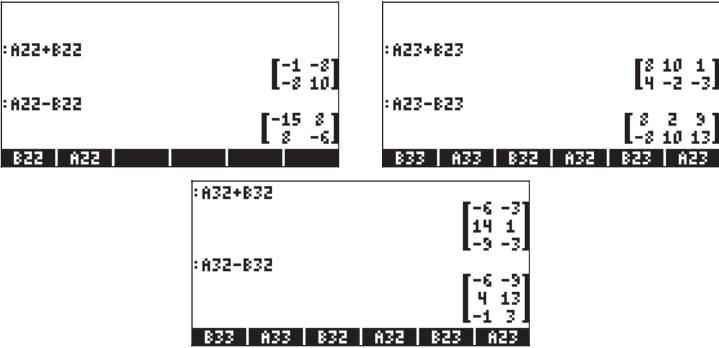
<pre>:RANM(2 2)>A22</pre> $\begin{bmatrix} -8 & 0 \\ 0 & 2 \end{bmatrix}$ <pre>:RANM(2 2)>B22</pre> $\begin{bmatrix} 7 & -8 \\ -8 & 8 \end{bmatrix}$ <pre>B22 A22 </pre> <pre>:RANM(3 2)>A32</pre> $\begin{bmatrix} -6 & -6 \\ 9 & 7 \\ -5 & 0 \end{bmatrix}$ <pre>:RANM(3 2)>B32</pre> $\begin{bmatrix} 0 & 3 \\ 5 & -6 \\ -4 & -3 \end{bmatrix}$ <pre>B32 A32 B23 A23 B22 A22 </pre>	<pre>:RANM(2 3)>A23</pre> $\begin{bmatrix} 8 & 6 & 5 \\ -2 & 4 & 5 \end{bmatrix}$ <pre>:RANM(2 3)>B23</pre> $\begin{bmatrix} 0 & 4 & -4 \\ 6 & -6 & -8 \end{bmatrix}$ <pre>B23 A23 B22 A22 </pre> <pre>:RANM(3 3)>A33</pre> $\begin{bmatrix} -8 & -3 & 4 \\ 7 & 8 & 6 \\ 5 & -1 & 4 \end{bmatrix}$ <pre>:RANM(3 3)>B33</pre> $\begin{bmatrix} -4 & 1 & 7 \\ -4 & -5 & 7 \\ -7 & 6 & 2 \end{bmatrix}$ <pre>B33 A33 B32 A32 B23 A23 </pre>
--	--

In RPN mode, the steps to follow are:

(2,2) ENTER RANM 'A22' STO►	(2,2) ENTER RANM 'B22' STO►
(2,3) ENTER RANM 'A23' STO►	(2,3) ENTER RANM 'B23' STO►
(3,2) ENTER RANM 'A32' STO►	(3,2) ENTER RANM 'B32' STO►
(3,3) ENTER RANM 'A33' STO►	(3,3) ENTER RANM 'B33' STO►

Addition and subtraction

Consider a pair of matrices $\mathbf{A} = [a_{ij}]_{m \times n}$ and $\mathbf{B} = [b_{ij}]_{m \times n}$. Addition and subtraction of these two matrices is only possible if they have the same number of rows and columns. The resulting matrix, $\mathbf{C} = \mathbf{A} \pm \mathbf{B} = [c_{ij}]_{m \times n}$ has elements $c_{ij} = a_{ij} \pm b_{ij}$. Some examples in ALG mode are shown below using the matrices stored above (e.g.,  \pm )



In RPN mode, the steps to follow are:

A_{22} ENTER B_{22} ENTER $+$ A_{22} ENTER B_{22} ENTER $-$
 A_{23} ENTER B_{23} ENTER $+$ A_{23} ENTER B_{23} ENTER $-$
 A_{32} ENTER B_{32} ENTER $+$ A_{32} ENTER B_{32} ENTER $-$

Translating the ALG examples to RPN is straightforward, as illustrated here. The remaining examples of matrix operations will be performed in ALG mode only.

Multiplication

There are numerous multiplication operations that involve matrices. These are described next.

Multiplication by a scalar

Multiplication of the matrix $\mathbf{A} = [a_{ij}]_{m \times n}$ by a scalar k results in the matrix $\mathbf{C} = k\mathbf{A} = [c_{ij}]_{m \times n} = [ka_{ij}]_{m \times n}$. In particular, the negative of a matrix is defined by the operation $-\mathbf{A} = (-1)\mathbf{A} = [-a_{ij}]_{m \times n}$. Some examples of multiplication of a matrix by a scalar are shown below.

```

:5*A22
      [-30 -30]
      [45 35]
      [-25 0]

:-2*B33
      [32 -2 -56]
      [32 40 -56]
      [56 -48 -16]

B23 | A33 | B32 | A32 | B23 | A23

```

```

      [22 40 -56]
      [56 -48 -16]

:-B23
      [0 -4 4]
      [-6 6 2]

:1.25*A22
      [-10. 0]
      [0 2.5]

B22 | A22 |

```

By combining addition and subtraction with multiplication by a scalar we can form linear combinations of matrices of the same dimensions, e.g.,

```

:5*A33-6*B33
      [-16 -21 -22]
      [59 70 -12]
      [67 -41 8]

:-3*B23-7*A23
      [-56 -54 -23]
      [-4 -10 -11]

B33 | A33 | B32 | A32 | B23 | A23

```

```

:2*A22-9*B22
      [-79 72]
      [72 -68]

:5*A32-n*B32
      [-30 -30-3n]
      [45-5n 35-6n]
      [-25-4n -(-3n)]

B33 | A33 | B32 | A32 | B23 | A23

```

In a linear combination of matrices, we can multiply a matrix by an imaginary number to obtain a matrix of complex numbers, e.g.,

```

RAD XYZ HEX C= 'X'      ALG
CHONE2
:2*A33-6*i*B33
      [-16--4*6i -6-6i 8-7*6i]
      [14--4*6i 16--5*6i 12-7*6i]
      [10--7*6i -2-6*6i 8-2*6i]
:EXPAND(ANS(1))
      [-(16-24i) -(6+6i) 8-42i]
      [14+24i 16+30i 12-42i]
      [10+42i -(2+36i) 8-12i]
COLLEXPANFACTO LDCOL L10 PARTF

```

Matrix-vector multiplication

Matrix-vector multiplication is possible only if the number of columns of the matrix is equal to the length of the vector. This operation follows the rules of matrix multiplication as shown in the next section. A couple of examples of matrix-vector multiplication follow:

```

      [-4 1 7]
      [-4 -5 7]
      [-7 6 2]

:ANS(1)*C1 2 -3J
      C-23 -35 -17

B23 | A33 | B32 | A32 | B23 | A23

```

```

:B32
      [0 3]
      [5 -6]
      [-4 -3]

:ANS(1)*C1 -2J
      C-6 17 23

B33 | A33 | B32 | A32 | B23 | A23

```

Vector-matrix multiplication, on the other hand, is not defined. This multiplication can be performed, however, as a special case of matrix multiplication as defined next.

Matrix multiplication

Matrix multiplication is defined by $\mathbf{C}_{m \times n} = \mathbf{A}_{m \times p} \cdot \mathbf{B}_{p \times n}$, where $\mathbf{A} = [a_{ij}]_{m \times p}$, $\mathbf{B} = [b_{ij}]_{p \times n}$, and $\mathbf{C} = [c_{ij}]_{m \times n}$. Notice that matrix multiplication is only possible if the number of columns in the first operand is equal to the number of rows of the second operand. The general term in the product, c_{ij} , is defined as

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

This is the same as saying that the element in the i -th row and j -th column of the product, \mathbf{C} , results from multiplying term-by-term the i -th row of \mathbf{A} with the j -th column of \mathbf{B} , and adding the products together. Matrix multiplication is not commutative, i.e., in general, $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$. Furthermore, one of the multiplications may not even exist.

The following screen shots show the results of multiplications of the matrices that we stored earlier:

<pre> :A33*B33 [16 31 -63] [-102 3 117] [-44 34 36] :B33*A33 [74 13 18] [32 -35 -18] [108 67 16] </pre>	<pre> :A22*B22 [-56 64] [-16 16] :B22*A22 [-56 -16] [64 16] </pre>
<pre> :B32*A23 [-6 12 15] [52 6 -5] [-26 -36 -35] :B23*A32 [56 28] [-50 -78] </pre>	<pre> :B32*B22 [-24 24] [23 -28] [-4 8] :B23*A33 [8 36 8] [-130 -58 -44] </pre>

The matrix-vector multiplication introduced in the previous section can be thought of as the product of a matrix $m \times n$ with a matrix $n \times 1$ (i.e., a column vector) resulting in an $m \times 1$ matrix (i.e., another vector). To verify this assertion check the examples presented in the previous section. Thus, the vectors defined in Chapter 9 are basically column vectors for the purpose of matrix multiplication.

The product of a vector with a matrix is possible if the vector is a row vector, i.e., a $1 \times m$ matrix, which multiplied with a matrix $m \times n$ produces a $1 \times n$ matrix

(another row vector). For the calculator to identify a row vector, you must use double brackets to enter it:

```
:B22
      [ 0 3 ]
      [ 5 -6 ]
      [-4 -3 ]
: [1 3 6] * ANS(1)
      [-9 -33]
B22 | A22 | B22 | A22 | B22 | A22
```

```
:B23
      [ 0 4 -4 ]
      [ 6 -6 -8 ]
: [1 3] * ANS(1)
      [18 -14 -28]
B23 | A23 | B23 | A23 | B23 | A23
```

Term-by-term multiplication

Term-by-term multiplication of two matrices of the same dimensions is possible through the use of function HADAMARD. The result is, of course, another matrix of the same dimensions. This function is available through Function catalog (\rightarrow CAT), or through the MATRICES/OPERATIONS sub-menu (\leftarrow MATRICES).

Applications of function HADAMARD are presented next:

```
:HADAMARD(A23,B23)
      [ 32 -3 28 ]
      [-28 -40 42 ]
      [-35 -6 8 ]
:HADAMARD(A22,B22)
      [-56 0 ]
      [ 0 16 ]
B22 | A22 | B22 | A22 | B22 | A22
```

```
:HADAMARD(B22,A22)
      [ 0 -18 ]
      [ 45 -42 ]
      [ 20 0 ]
:HADAMARD(B23,A23)
      [ 0 24 -20 ]
      [-12 -24 -40]
B23 | A23 | B23 | A23 | B23 | A23
```

Raising a matrix to a real power

You can raise a matrix to any power as long as the power is either an integer or a real number with no fractional part. The following example shows the result of raising matrix B22, created earlier, to the power of 5:

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

You can also raise a matrix to a power without first storing it as a variable:

$$: \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^4 \quad \begin{bmatrix} 199 & 290 \\ 435 & 634 \end{bmatrix}$$

In algebraic mode, the keystrokes are: [enter or select the matrix] $\boxed{Y^x}$ [enter the power] $\boxed{\text{ENTER}}$. In RPN mode, the keystrokes are: [enter or select the matrix] $\boxed{\text{SPC}}$ [enter the power] $\boxed{Y^x}$ $\boxed{\text{ENTER}}$.

Matrices can be raised to negative powers. In this case, the result is equivalent to $1/[\text{matrix}]^{\text{ABS}(\text{power})}$.

$$: \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{(-4)} \quad \begin{bmatrix} 317 & -145 \\ 8 & 8 \\ -435 & 199 \\ 16 & 16 \end{bmatrix}$$

The identity matrix

In Chapter 9 we introduce the identity matrix as the matrix $\mathbf{I} = [\delta_{ij}]_{n \times n}$, where δ_{ij} is the Kronecker's delta function. Identity matrices can be obtained by using function IDN described in Chapter 9. The identity matrix has the property that $\mathbf{A} \cdot \mathbf{I} = \mathbf{I} \cdot \mathbf{A} = \mathbf{A}$. To verify this property we present the following examples using the matrices stored earlier on:

$$: A22 \quad \begin{bmatrix} -8 & 0 \\ 0 & 2 \end{bmatrix}$$

$$: A22 \cdot \text{IDN}(A22) \quad \begin{bmatrix} -8 & 0 \\ 0 & 2 \end{bmatrix}$$

The inverse matrix

The inverse of a square matrix \mathbf{A} is the matrix \mathbf{A}^{-1} such that $\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{I}$, where \mathbf{I} is the identity matrix of the same dimensions as \mathbf{A} . The inverse of a matrix is obtained in the calculator by using the inverse function, INV (i.e., the $\boxed{1/x}$ key). An example of the inverse of one of the matrices stored earlier is presented next:

: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

To verify the properties of the inverse matrix, consider the following multiplications:

: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	1
			0	1	1
			1	0	1
B22	A22				

Characterizing a matrix (The matrix NORM menu)

The matrix NORM (NORMALIZE) menu is accessed through the keystroke sequence \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

These functions are described next. Because many of these functions use concepts of matrix theory, such as singular values, rank, etc., we will include short descriptions of these concepts intermingled with the description of functions.

Function ABS

Function ABS calculates what is known as the Frobenius norm of a matrix. For a matrix $\mathbf{A} = [a_{ij}]_{m \times n}$, the Frobenius norm of the matrix is defined as

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

If the matrix under consideration is a row vector or a column vector, then the Frobenius norm, $\|\mathbf{A}\|_F$, is simply the vector's magnitude. Function ABS is accessible directly in the keyboard as  ABS.

Try the following exercises in ALG mode (using the matrices stored earlier for matrix operations):

: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

Function SNRM calculates the Spectral NoRM of a matrix, which is defined as the matrix's largest singular value, also known as the Euclidean norm of the matrix. For example,

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

Singular value decomposition

To understand the operation of Function SNRM, we need to introduce the concept of matrix decomposition. Basically, matrix decomposition involves the determination of two or more matrices that, when multiplied in a certain order (and, perhaps, with some matrix inversion or transposition thrown in), produce the original matrix. The Singular Value Decomposition (SVD) is such that a rectangular matrix $\mathbf{A}_{m \times n}$ is written as $\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \cdot \mathbf{S}_{m \times n} \cdot \mathbf{V}_{n \times n}^T$ 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).

The rank of a matrix can be determined from its SVD by counting the number of non-singular values. Examples of SVD will be presented in a subsequent section.

Functions RNRM and CNRM

Function RNRM returns the Row NoRM of a matrix, while function CNRM returns the Column NoRM of a matrix. Examples,

```
: RNRM(A22)
: CNRM(A22)
: RNRM(A33)
21
ABS | SNRM | RNRM | CNRM | SRAD | COND
```

```
: CNRM(A33)
: RNRM(A23)
: CNRM(A23)
21
20
19
10
ABS | SNRM | RNRM | CNRM | SRAD | COND
```

Row norm and column norm of a matrix

The row norm of a matrix is calculated by taking the sums of the absolute values of all elements in each row, and then, selecting the maximum of these sums. The column norm of a matrix is calculated by taking the sums of the absolute values of all elements in each column, and then, selecting the maximum of these sums.

Function SRAD

Function SRAD determines the Spectral RADIUS of a matrix, defined as the largest of the absolute values of its eigenvalues. For example,

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

Definition of eigenvalues and eigenvectors of a matrix

The eigenvalues of a square matrix result from the matrix equation $\mathbf{A} \cdot \mathbf{x} = \lambda \cdot \mathbf{x}$. The values of λ that satisfy the equation are known as the eigenvalues of the matrix \mathbf{A} . The values of \mathbf{x} that result from the equation for each value of λ are known as the eigenvectors of the matrix. Further details on calculating eigenvalues and eigenvectors are presented later in the chapter.

Function COND

Function COND determines the condition number of a matrix:

```
: COND(A22)
                                     4.
: COND(B33)
      9.88617886179
: COND(A33)
      6.78714859438
A | X | B33 | A33 | B32 | A32
```

Condition number of a matrix

The condition number of a square non-singular matrix is defined as the product of the matrix norm times the norm of its inverse, i.e.,

$\text{cond}(\mathbf{A}) = ||\mathbf{A}|| \times ||\mathbf{A}^{-1}||$. We will choose as the matrix norm, $||\mathbf{A}||$, the maximum of its row norm (RNRN) and column norm (CNRN), while the norm of the inverse, $||\mathbf{A}^{-1}||$, will be selected as the minimum of its row norm and column norm. Thus, $||\mathbf{A}|| = \max(\text{RNRN}(\mathbf{A}), \text{CNRN}(\mathbf{A}))$, and $||\mathbf{A}^{-1}|| = \min(\text{RNRN}(\mathbf{A}^{-1}), \text{CNRN}(\mathbf{A}^{-1}))$.

The condition number of a singular matrix is infinity. The condition number of a non-singular matrix is a measure of how close the matrix is to being singular. The larger the value of the condition number, the closer it is to singularity. (A singular matrix is one for which the inverse does not exist).

Try the following exercise for matrix condition number on matrix A33. The condition number is COND(A33) , row norm, and column norm for A33 are shown to the left. The corresponding numbers for the inverse matrix, INV(A33), are shown to the right:

: COND(A33)	6.78714859438	: COND(INV(A33))	20.
: RNRM(A33)	21.	: RNRM(INV(A33))	6.78714859437
: CNRM(A33)	20.	: CNRM(INV(A33))	.261044176707
			.339357429719
HADAM	LSQ	HAD	RANK
RNRM	ASD	RNRM	ASD

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

Function RANK

Function RANK determines the rank of a square matrix. Try the following examples:

: RANK(A22)	2
: RANK(B22)	2
B22	A22
PPAR	A
F	X

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

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

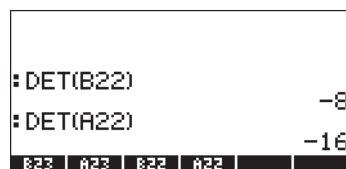
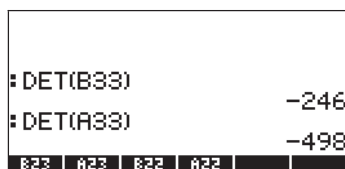
For example, try finding the rank for the matrix:



You will find that the rank is 2. That is because the second row $[2,4,6]$ is equal to the first row $[1,2,3]$ multiplied by 2, thus, row two is linearly dependent of row 1 and the maximum number of linearly independent rows is 2. You can check that the maximum number of linearly independent columns is 3. The rank being the maximum number of linearly independent rows or columns becomes 2 for this case.

Function DET

Function DET calculates the determinant of a square matrix. For example,

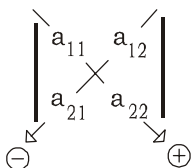


The determinant of a matrix

The determinant of a 2x2 and or a 3x3 matrix are represented by the same arrangement of elements of the matrices, but enclosed between vertical lines, i.e.,

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

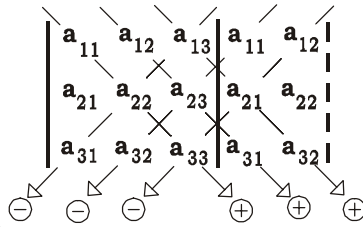
A 2x2 determinant is calculated by multiplying the elements in its diagonal and adding those products accompanied by the positive or negative sign as indicated in the diagram shown below.



The 2x2 determinant is, therefore,

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

A 3x3 determinant is calculated by *augmenting* the determinant, an operation that consists on copying the first two columns of the determinant, and placing them to the right of column 3, as shown in the diagram below. The diagram also shows the elements to be multiplied with the corresponding sign to attach to their product, in a similar fashion as done earlier for a 2x2 determinant. After multiplication the results are added together to obtain the determinant.



For square matrices of higher order determinants can be calculated by using smaller order determinant called cofactors. The general idea is to "expand" a determinant of a $n \times n$ matrix (also referred to as a $n \times n$ determinant) into a sum of the cofactors, which are $(n-1) \times (n-1)$ determinants, multiplied by the elements of a single row or column, with alternating positive and negative signs. This "expansion" is then carried to the next (lower) level, with cofactors of order $(n-2) \times (n-2)$, and so on, until we are left only with a long sum of 2×2 determinants. The 2×2 determinants are then calculated through the method shown above.

The method of calculating a determinant by cofactor expansion is very inefficient in the sense that it involves a number of operations that grows very fast as the size of the determinant increases. A more efficient method, and the one preferred in numerical applications, is to use a result from Gaussian elimination. The method of Gaussian elimination is used to solve systems of linear equations. Details of this method are presented in a later part of this chapter.

To refer to the determinant of a matrix \mathbf{A} , we write $\det(\mathbf{A})$. A singular matrix has a determinant equal to zero.

Function TRACE

Function TRACE calculates the trace of square matrix, defined as the sum of the elements in its main diagonal, or

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

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

Function TRAN returns the transpose of a real or the conjugate transpose of a complex matrix. TRAN is equivalent to TRN. The operation of function TRN was presented in Chapter 10.

Additional matrix operations (The matrix OPER menu)

The matrix OPER (OPERATIONS) is available through the keystroke sequence

MATRICES (system flag 117 set to CHOOSE boxes):



The OPERATIONS menu includes the following functions:

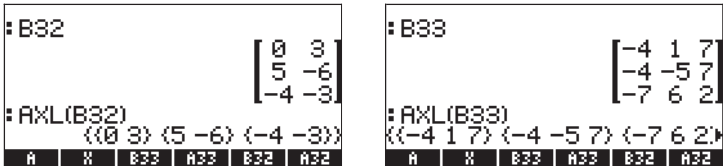


Functions ABS, CNRM, COND, DET, RANK, RNRM, SNRM, TRACE, and TRAN are also found in the MTH/MATRIX/NORM menu (the subject of the previous section). Function SIZE was presented in Chapter 10. Function HADAMARD was presented earlier in the context of matrix multiplication. Functions LSQ ,

MAD and RSD are related to the solution of systems of linear equations and will be presented in a subsequent section in this Chapter. In this section we'll discuss only functions AXL and AXM.

Function AXL

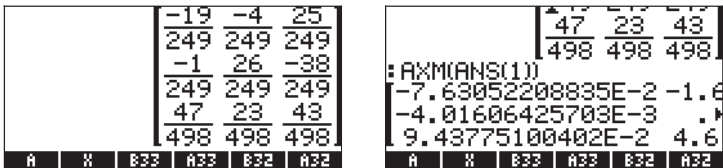
Function AXL converts an array (matrix) into a list, and vice versa:



Note: the latter operation is similar to that of the program CRMR presented in Chapter 10.

Function AXM

Function AXM converts an array containing integer or fraction elements into its corresponding decimal, or approximate, form:



Function LCXM

Function LCXM can be used to generate matrices such that the element a_{ij} is a function of i and j . The input to this function consists of two integers, n and m , representing the number of rows and columns of the matrix to be generated, and a program that takes i and j as input. The numbers n , m , and the program occupy stack levels 3, 2, and 1, respectively. Function LCXM is accessible through the command catalog CAT .

For example, to generate a 2' 3 matrix whose elements are given by $a_{ij} = (i+j)^2$, first, store the following program into variable P1 in RPN mode. This is the way that the RPN stack looks before pressing .


```

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

```

The implementation of function LEXM for this case requires you to enter:

2 **ENTER** **3** **ENTER** **→** **LCXM** **ENTER**

The following figure shows the RPN stack before and after applying function LEXM:

0:	2:
1:	1:
« ÷ i j « '(i+j)^2.	
' EVAL » »	[4. 9. 16.]
P1 B33 A33 B23 A23 B22	P1 B33 A33 B23 A23 B22

In ALG mode, this example can be obtained by using:

```

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

```

The program P1 must still have been created and stored in RPN mode.

Solution of linear systems

A system of n linear equations in m variables can be written as

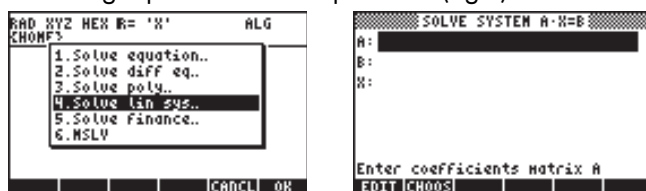
$$\begin{aligned}
 a_{11} \cdot x_1 + a_{12} \cdot x_2 + a_{13} \cdot x_3 + \dots + a_{1,m-1} \cdot x_{m-1} + a_{1,m} \cdot x_m &= b_1, \\
 a_{21} \cdot x_1 + a_{22} \cdot x_2 + a_{23} \cdot x_3 + \dots + a_{2,m-1} \cdot x_{m-1} + a_{2,m} \cdot x_m &= b_2, \\
 a_{31} \cdot x_1 + a_{32} \cdot x_2 + a_{33}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

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.

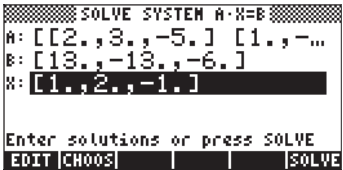
To enter matrix **A** you can activate the Matrix Writer while the A: field is selected. The following screen shows the Matrix Writer used for entering matrix **A**, as well as the input form for the numerical solver after entering matrix **A** (press **ENTER** in the Matrix Writer):



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.]
 B33 | A33 | B32 | A32 | B23 | A23

To check that the solution is correct, enter the matrix A and multiply times this solution vector (example in algebraic mode):

Solutions:[1. 2. -1.]
 : [2 3 -5] *ANS(1)
 [1 -3 8]
 [2 -2 4] [13. -13. -6.]
 B33 | A33 | B32 | A32 | B23 | A23

Under-determined system

The system of linear equations

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







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

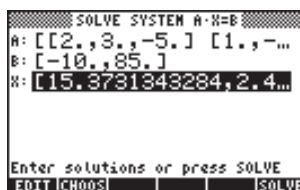
can be written as the matrix equation $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, if

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

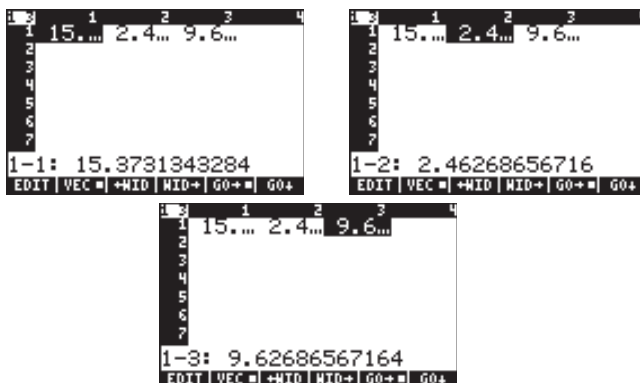
This system has more unknowns than equations, therefore, it is not uniquely determined. We can visualize the meaning of this statement by realizing that each of the linear equations represents a plane in the three-dimensional Cartesian coordinate system (x_1, x_2, x_3) . The solution to the system of equations shown above will be the intersection of two planes in space. We know, however, that the intersection of two (non-parallel) planes is a straight line, and not a single point. Therefore, there is more than one point that satisfy the system. In that sense, the system is not uniquely determined.

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

 NUM.SLV     . 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 .

The procedure that we describe next can be used to copy the matrix A and the solution vector X into the stack. To check that the solution is correct, try the following:

- Press , to highlight the A: field.
- Press , to copy matrix A onto the stack.
- Press to return to the numerical solver environment.
- Press , to copy solution vector X onto the stack.
- Press to return to the numerical solver environment.
- Press to return to the stack.

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

Solutions: [15.37313432
[2.3, -5.]
[1. -3. 8.]
[15.3731343284 2.46268
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 $\text{STO} \rightarrow$ ALPHA X ENTER to store the solution vector into variable X

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

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

Now, let's verify the solution by using: $\text{[2D]} \times \text{[2D]} \text{ENTER}$, which results in (press ∇ to see the vector elements): [-9.99999999992 85.], close enough to the original vector $\mathbf{b} = [-10 \ 85]$.

Try also this, $\text{[2D]} \times [15, 10/3, 10] \text{ENTER} \rightarrow \text{NUM} \text{ENTER}$, i.e.,

:A*[15 10/3 10]
[-30. 255.]
:NUM(A*[15 10/3 10])
[-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

$$x_1 + 3x_2 = 15,$$

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







$$-x_1 + x_2 = 22,$$

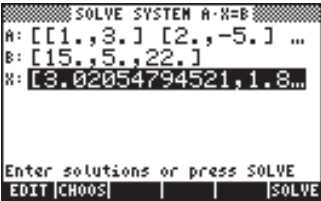
can be written as the matrix equation $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, if


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

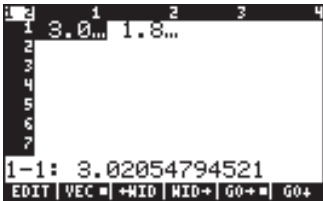
This system has more equations than unknowns (an over-determined system). The system does not have a single solution. Each of the linear equations in the system presented above represents a straight line in a two-dimensional Cartesian coordinate system (x_1, x_2). Unless two of the three equations in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



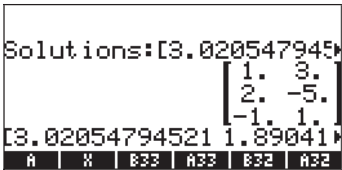
To see the details of the solution vector, if needed, press the  button. This will activate the Matrix Writer. Within this environment, use the right- and left-arrow keys to move about the vector:



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 **▢** to return to the numerical solver environment.
- Press **↓** **↓** **▢** **ENTER** , to copy solution vector X onto the stack.
- Press **▢** 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.

Least-square solution (function LSQ)

The LSQ function returns the minimum-norm least-square solution of a linear system $Ax = b$, according to the following criteria:

- If **A** is a square matrix and **A** is non-singular (i.e., its inverse matrix exist, or its determinant is non-zero), LSQ returns the exact solution to the linear system.
- If **A** has less than full row rank (underdetermined system of equations), LSQ returns the solution with the minimum Euclidean length out of an infinity number of solutions.
- If **A** has less than full column rank (over-determined system of equations), LSQ returns the "solution" with the minimum residual value $\mathbf{e} = \mathbf{A} \cdot \mathbf{x} - \mathbf{b}$. The system of equations may not have a solution, therefore, the value returned is not a real solution to the system, just the one with the smallest residual.

Function LSQ takes as input vector **b** and matrix **A**, in that order. Function LSQ can be found in Function catalog ( CAT). Next, we use function LSQ to repeat the solutions found earlier with the numerical solver:

Square system

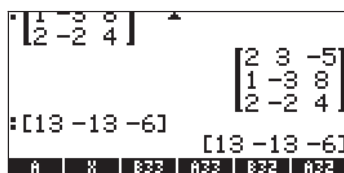
Consider the system

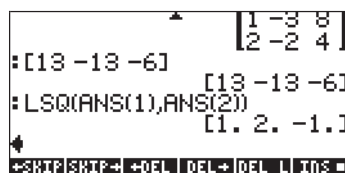
$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= 13, \\ x_1 - 3x_2 + 8x_3 &= -13, \\ 2x_1 - 2x_2 + 4x_3 &= -6, \end{aligned}$$

with

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

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:

Compare these three solutions with the ones calculated with the numerical solver.

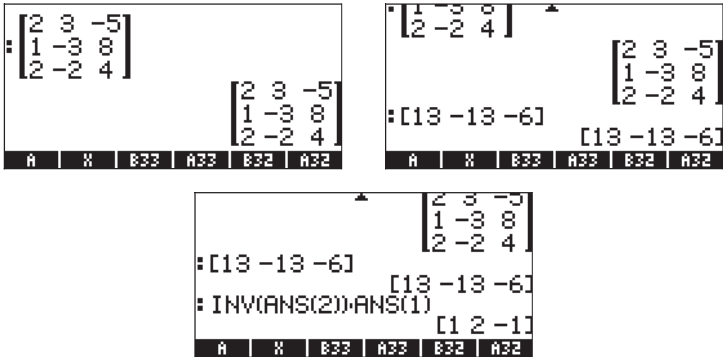
Solution with the inverse matrix

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:



which is the same result found earlier.

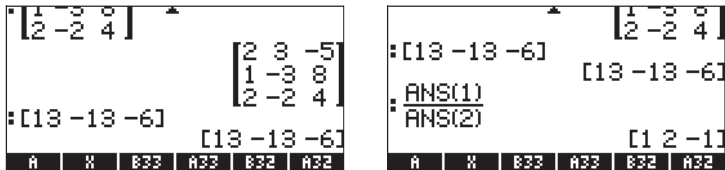
Solution by “division” of matrices

While the operation of division is not defined for matrices, we can use the calculator’s \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.

The procedure for the case of “dividing” **b** by **A** is illustrated below for the case

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

The procedure is shown in the following screen shots:



The same solution as found above with the inverse matrix.

Solving multiple set of equations with the same coefficient matrix

Suppose that you want to solve the following three sets of equations:

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

We can write the three systems of equations as a single matrix equation: **A**·**X** = **B**, where

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

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

The sub-indices in the variable names X, Y, and Z, determine to which equation system they refer to. To solve this expanded system we use the following procedure, in RPN mode,

```
[[14,9,-2],[2,-5,2],[5,19,12]] ENTER
[[1,2,3],[3,-2,1],[4,2,-1]] ENTER ÷
```

The result of this operation is:

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

Gaussian and Gauss-Jordan elimination

Gaussian elimination is a procedure by which the square matrix of coefficients belonging to a system of n linear equations in n unknowns is reduced to an upper-triangular matrix (*echelon form*) through a series of row operations. This procedure is known as *forward elimination*. The reduction of the coefficient matrix to an upper-triangular form allows for the solution of all n unknowns, utilizing only one equation at a time, in a procedure known as *backward substitution*.

Example of Gaussian elimination using equations

To illustrate the Gaussian elimination procedure we will use the following system of 3 equations in 3 unknowns:

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

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

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

We can store these equations in the calculator in variables E1, E2, and E3, respectively, as shown below. For backup purposes, a list containing the three equations was also created and stored into variable EQS. This way, if a mistake is made, the equations will still be available to the user.

```
: 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 | E3 | E2 | E1 |
```

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

To start the process of forward elimination, we divide the first equation (E1) by 2, and store it in E1, and show the three equations again to produce:

: E1	2	→ E1			
			X+2Y+3Z=7		
: E2					
			3X-(2Y-Z)=-8		
: E3					
			4X+2Y-Z=-4		
E05	E3	E2	E1		

Next, we replace the second equation E2 by (equation 2 – 3×equation 1, i.e., E1-3×E2), and the third by (equation 3 – 4×equation 1), to get:

: E2-3·E1	→ E2				
			-(8Y+8Z-24)		
: E3-4·E1	→ E3				
			-(6Y+13Z-32)		
E05	E3	E2	E1		

: E1			X+2Y+3Z=7		
: E2			-(8Y+8Z-24)		
: E3			-(6Y+13Z-32)		
E05	E3	E2	E1		

Next, divide the second equation by -8, to get:

: E2	-8	→ E2			
			Y+Z=3		
E05	E3	E2	E1		

: E2	-8	→ E2			
			Y+Z=3		
E05	E3	E2	E1		

Next, replace the third equation, E3, with (equation 3 + 6×equation 2, i.e., E2+6×E3), to get:

: E3+6·E2	→ E3				
			-(7Z-14)		
E05	E3	E2	E1		

: E1			X+2Y+3Z=7		
: E2			Y+Z=3		
: E3			-(7Z-14)		
E05	E3	E2	E1		

Notice that when we perform a linear combination of equations the calculator modifies the result to an expression on the left-hand side of the equal sign, i.e.,

an expression = 0. Thus, the last set of equations is interpreted to be the following equivalent set of equations:

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

The process of backward substitution in Gaussian elimination consists in finding the values of the unknowns, starting from the last equation and working upwards. Thus, we solve for Z first:

X+2Y+3Z=7		Y+Z=3	
: E2		: E3	
	Y+Z=3		-(7Z-14)
: E3		: SOLVE(E3,'Z')	
	-(7Z-14)		Z=2
: SOLVE(E3,'Z')		: SUBST(E2,ANS(1))►E2	
	Z=2		Y+2=3
EQS	E3	E2	E1

Next, we substitute Z=2 into equation 2 (E2), and solve E2 for Y:

-(7Z-14)	
: SOLVE(E3,'Z')	Z=2
: SUBST(E2,ANS(1))►E2	Y+2=3
: SOLVE(E2,'Y')	Y=1
EQS	E3

Next, we substitute Z=2 and Y = 1 into E1, and solve E1 for X:

Y=1		X+2Y+3Z=7	
: SUBST(E1,Y=1)		: SUBST(ANS(1),Z=2)	
	X+2Y+3Z=7		X+2Y+3Z=7
: SUBST(ANS(1),Z=2)		: ANS(1)►E1	
	X+2Y+3Z=7		X+2Y+3Z=7
: ANS(1)►E1		: SOLVE(ANS(1),'X')	
	X+2Y+3Z=7		X=-1
EQS	E3	E2	E1

The solution is, therefore, X = -1, Y = 1, Z = 2.

Example of Gaussian elimination using matrices

The system of equations used in the example above can be written as a matrix equation $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, if we use:

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

To obtain a solution to the system matrix equation using Gaussian elimination, we first create what is known as the augmented matrix corresponding to \mathbf{A} , i.e.,

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

The matrix \mathbf{A}_{aug} is the same as the original matrix \mathbf{A} with a new row, corresponding to the elements of the vector \mathbf{b} , added (i.e., augmented) to the right of the rightmost column of \mathbf{A} .

Once the augmented matrix is put together, we can proceed to perform row operations on it that will reduce the original A matrix into an upper-triangular matrix. For this exercise, we will use the RPN mode (MODE \rightarrow +/- \rightarrow \rightarrow), with system flag 117 set to SOFT menu. In your calculator, use the following keystrokes. First, enter the augmented matrix, and make an extra copy of the same in the stack (This step is not necessary, except as an insurance that you have an extra copy of the augmented matrix saved in case you make a mistake in the forward elimination procedure that we are about to undertake.):

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

Save augmented matrix in variable AAUG: `[, ALPHA ALPHA A A U G ALPHA STOP`

With a copy of the augmented matrix in the stack, press \leftarrow \overline{MTH} $\left[\begin{smallmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{smallmatrix}\right]$ $\left[\begin{smallmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{smallmatrix}\right]$ to activate the ROW operation menu. Next, perform the following row operations on your augmented matrix.

Multiply row 1 by $\frac{1}{2}$:

Multiply row 1 by -3 add it to row 2, replacing it:

Multiply row 1 by -4 add it to row 3, replacing it:

Multiply row 2 by $-1/8$:

Multiply row 2 by 6 add it to row 3, replacing it:

If you were performing these operations by hand, you would write the following:

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

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

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

The symbol \cong ("is equivalent to") indicates that what follows is equivalent to the previous matrix with some row (or column) operations involved.

The resulting matrix is upper-triangular, and equivalent to the set of equations

$$X + 2Y + 3Z = 7,$$


$$Y + Z = 3,$$

$$-7Z = -14,$$

which can now be solved, one equation at a time, by backward substitution, as in the previous example.

Gauss-Jordan elimination using matrices


Gauss-Jordan elimination consists in continuing the row operations in the upper-triangular matrix resulting from the forward elimination process until an identity matrix results in place of the original \mathbf{A} matrix. For example, for the case we just presented, we can continue the row operations as follows:

7 +/- 1/x 3 

Multiply row 3 by -1 , add it to row 2, replacing it:

SPC 2 333

Multiply row 3 by -3 , add it to row 1, replacing it:

3 +/- SPC 3 SPC / 

Multiply row 2 by -2 , add it to row 1, replacing it:



Writing this process by hand will result in the following steps:

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

$$A_{aug} \equiv \left(\begin{array}{ccc|c} 1 & 2 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array} \right) \equiv \left(\begin{array}{ccc|c} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array} \right).$$

Pivoting

If you look carefully at the row operations in the examples shown above, you will notice that many of those operations divide a row by its corresponding element in the main diagonal. This element is called a pivot element, or simply, a pivot. In many situations it is possible that the pivot element become zero, in which case we cannot divide the row by its pivot. Also, to improve the numerical solution of a system of equations using Gaussian or Gauss-Jordan elimination, it is recommended that the pivot be the element with the largest absolute value in a given column. In such cases, we exchange rows before performing row operations. This exchange of rows is called partial pivoting. To follow this recommendation is it often necessary to exchange rows in the augmented matrix while performing a Gaussian or Gauss-Jordan elimination.

While performing pivoting in a matrix elimination procedure, you can improve the numerical solution even more by selecting as the pivot the element with the largest absolute value in the column and row of interest. This operation may require exchanging not only rows, but also columns, in some pivoting operations. When row and column exchanges are allowed in pivoting, the procedure is known as 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}_{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   to get a copy in the stack. We want to keep the CSWP (Column Swap) command readily available, for which we use:  CAT ALPHA ALPHA C S ALPHA (find CSWP), . You'll get an error message, press , and ignore the message. Next, get the ROW menu available by pressing:  MATRICES  .

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: 

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

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:




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

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




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

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




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

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: 


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

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

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

$$\left[\begin{array}{cccc} 1 & -1/16 & 0 & 33/16 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{array} \right] \quad \left[\begin{array}{ccc} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{array} \right]$$

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

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

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

We now have an identity matrix in the portion of the augmented matrix corresponding to the original coefficient matrix A , thus we can proceed to obtain the solution while accounting for the row and column exchanges coded in the permutation matrix P . We identify the unknown vector \mathbf{x} , the modified independent vector \mathbf{b}' and the permutation matrix P as:

$$\mathbf{x} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \mathbf{b}' = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

The solution is given by $P \cdot \mathbf{x} = \mathbf{b}'$, or

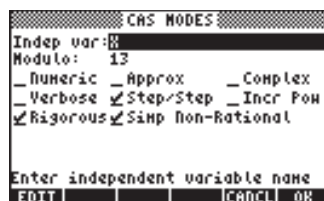
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix}.$$

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

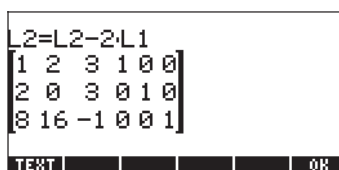
The example we just worked is, of course, the step-by-step, user-driven procedure to use full pivoting for Gauss-Jordan elimination solution of linear equation systems. You can see the step-by-step procedure used by the calculator to solve a system of equations, without user intervention, by setting the step-by-step option in the calculator's CAS, as follows:



Then, for this particular example, in RPN mode, use:

$[2, -1, 4]$ **ENTER** $[[1, 2, 3], [2, 0, 3], [8, 16, -1]]$ **ENTER** **÷**

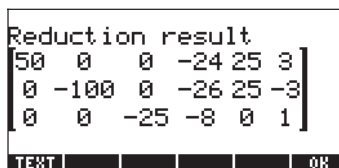
The calculator shows an augmented matrix consisting of the coefficients matrix **A** and the identity matrix **I**, while, at the same time, showing the next procedure to calculate:



$L2 = L2 - 2 \cdot L1$ stands for “replace row 2 ($L2$) with the operation $L2 - 2 \cdot L1$. If we had done this operation by hand, it would have corresponded to:

2 **+/-** **SPC** **/** **SPC** **/** **2nd** **DEL**. Press **OK**, and follow the operations in your calculator’s screen. You will see the following operations performed:

$L3 = L3 - 8 \cdot L1$, $L1 = 2 \cdot L1 - 1 \cdot L2$, $L1 = 25 \cdot L1 - 3 \cdot L3$, $L2 = 25 \cdot L2 - 3 \cdot L3$, and finally a message indicating “Reduction result” showing:



When you press **OK**, the calculator returns the final result $[1 \ 2 \ -1]$.

Calculating the inverse matrix step-by-step

The calculation of an inverse matrix can be considered as calculating the solution to the augmented system $[A \mid I]$. For example, for the matrix **A** used in the previous example, we would write this augmented matrix as

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

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

$\boxed{[[1,2,3],[3,-2,1],[4,2,-1]]}$ \boxed{ENTER} $\boxed{1/x}$

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

$$\boxed{\begin{array}{l} 1: \\ \left[\begin{array}{ccc} 0 & \frac{1}{7} & \frac{1}{7} \\ \frac{1}{8} & -\frac{13}{56} & \frac{1}{7} \\ \frac{1}{4} & \frac{3}{28} & -\frac{1}{7} \end{array} \right] \end{array}}$$

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

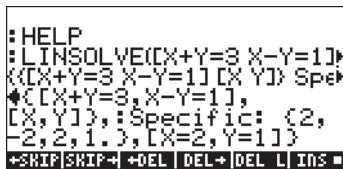
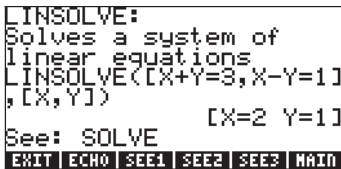
The simplest way to solve a system of linear equations, $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, in the calculator is to enter \mathbf{b} , enter \mathbf{A} , and then use the division function $/$. If the system of linear equations is over-determined or under-determined, a “solution” can be produced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

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

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.

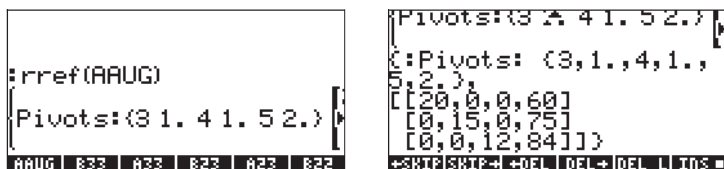
As an example, we show the result of applying function RREF to matrix AAUG in ALG mode:



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 ∇). 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])
: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 λ that satisfy the equation are known as the eigenvalues of matrix \mathbf{A} . For each value of λ , 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 λ 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:



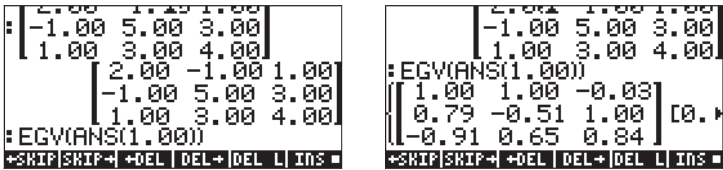
Change mode to Approx and repeat the entry, to get the following eigenvalues: $[(1.38, 2.22), (1.38, -2.22), (-1.76, 0)]$.

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

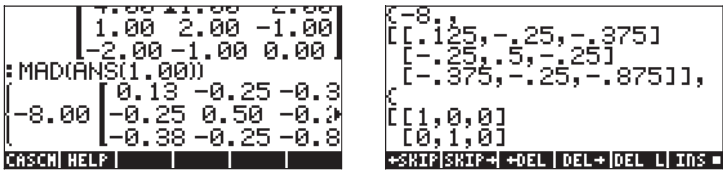
As an example, in RPN mode, try:

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

The result is:

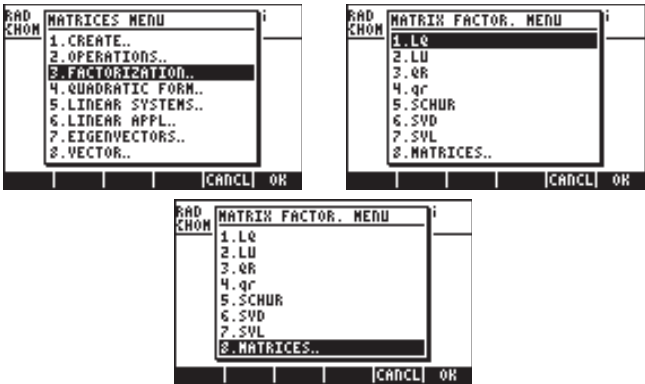
```
4: -8.  
3: [[0.13 -0.25 -0.38][-0.25 0.50 -0.25][-0.38 -0.25 -0.88]]  
2: {[[1 0 0][0 1 0][0 0 1]][[-2 1 -2][1 -4 -1][-2 -1 -6]][[-1 2 3][2 -4 2][3 2 7]]}  
1: 'X^3+6*x^2+2*X+8'
```

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.

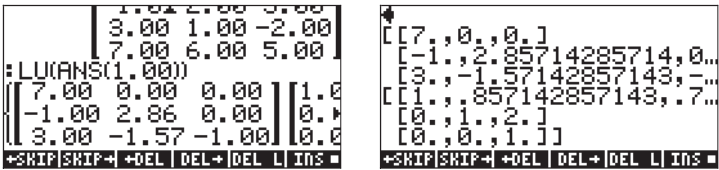
Function LU

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 δ_{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* × *m* matrix is the equivalent of a triangular matrix out of an *n* × *n* matrix). For example,

```
[[ 1, -2, 1][ 2, 1, -2][ 5, -2, 1]] LQ
```

produces

```
3: [[-5.48 0 0][-1.10 -2.79 0][-1.83 1.43 0.78]]
2: [[-0.91 0.37 -0.18][ -0.36 -0.50 0.79][ -0.20 -0.78 -0.59]]
1: [[0 0 1][0 1 0][1 0 0]]
```

Function QR

In RPN, function QR produces the *QR factorization* of a matrix $\mathbf{A}_{n \times m}$ returning a $\mathbf{Q}_{n \times n}$ orthogonal matrix, a $\mathbf{R}_{n \times m}$ upper trapezoidal matrix, and a $\mathbf{P}_{m \times m}$ permutation matrix, in stack levels 3, 2, and 1. The matrices \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 $\mathbf{x} = [X \ Y \ Z]^T$, the corresponding quadratic form is calculated as

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

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

Function SYLVESTER takes as argument a symmetric square matrix \mathbf{A} and returns a vector containing the diagonal terms of a diagonal matrix \mathbf{D} , and a matrix \mathbf{P} , so that $\mathbf{P}^T \cdot \mathbf{A} \cdot \mathbf{P} = \mathbf{D}$. For example:

```
[[2,1,-1],[1,4,2],[-1,2,-1]] SYLVESTER
```

produces

```
2: [ 1/2 2/7 -23/7]
```

```
1: [[2 1 -1][0 7/2 5/2][0 0 1]]
```

Function GAUSS

Function GAUSS returns the diagonal representation of a quadratic form $\mathbf{Q} = \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}^T$ 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 \mathbf{D} (stack level 4)
- A matrix \mathbf{P} such that $\mathbf{A} = \mathbf{P}^T \cdot \mathbf{D} \cdot \mathbf{P}$ (stack level 3)
- The diagonalized quadratic form (stack level 2)
- The list of variables (stack level 1)

For example:

```
'X^2+Y^2-Z^2+4*X*Y-16*X*Z' ENTER
['X','Y','Z'] ENTER GAUSS
```

returns


```
4: [1 -0.333 20.333]
```

```
3: [[1 2 -8][0 -3 16][0 0 1]]
```

```
2: '61/3*Z^2+ -1/3*(16*Z+3*Y)^2+(-8*z+2*Y+X)^2'
```

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

Linear Applications

The LINEAR APPLICATIONS menu is available through the  MATRICES .



Information on the functions listed in this menu is presented below by using the calculator's own help facility. The figures show the help facility entry and the attached examples.

Function IMAGE

```

IMAGE:
Image of a linear ap-
plication of matrix M
IMAGE([[1,2,3],[4,5,6]
])
      [[1 0] [0 1]]
See: KER BASIS
EXIT ECHO SEE1 SEE2 SEE3 MAIN

```

```

:HELP
: IMAGE([[1 2 3]
         [4 5 6]])
                        [[1 0] [0 1]]
CASCH 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}
CASCH 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
```