Vectores e matrices

MAPLE ten definidos internamente varios tipos de datos compostos: secuencias (ou secesións), conxuntos, listas, vectores, matrices e táboas.

Definición de vectores

Para definir un vector en MAPLE, utilízase o comando

Vector[o](n, init, ro, sym, sh, st, dt, f, a, o)

- **o** (opcional) especifica a orientación do vector ([row] horizontal e [column] vertical. Por defecto devolve un vector columna
- **n** (opcional) é un enteiro ou rango de enteiro especificando o número de elementos. Por defecto rechea o vector con 0.
- **init** (opcional) un procedemento MAPLE para especifica-los elementos iniciais do vector. Este procedemento pode ser a través dun tipo de dato table, array, list, Array, Matrix, Vector, conxunto de ecuacións ou expresións de tipo alxebráico.
- **ro** (opcional) Booleano. Especifica se o vector se pode modificar (false) ou non (true).
- **sym** (opcional) *symbol*=nome especifica o nome a utilizar nos elementos do vector cando non se inicializa o vector.
- **sh** (opcional) *shape*=nome ou *shape*=lista de nomes. Os posibeis nomes son: *constant[x]* (pon tódolos elementos ao valor *x*), *scalar[i, x]* (pon o elemento *i* ao valor *x*), *unit[i]* (pon a 0 tódolos elementos agás o *i*-ésimo, que pon a 1), *zero*.
- **dt** (opcional) *datatype*=nome especifica o tipo de dato almacenado no vector. Os posibeis nomes son: integer, float, complex.
- **f** (opcional) *fill*=valor, que é o valor de recheo do vector para os elementos restantes (por defecto 0).

```
> Vector(2); #vector con 2 elementos. Por defecto inicializado a 0.
```

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \textbf{(1.1.1)}$$

```
> Vector(1..3,5); #vector con 3 elementos inicializado a 5.
```

$$\begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix} \quad \textbf{(1.1.2)}$$

```
> Vector[row]([1, x^2+y, sqrt(2)]); #vector fila inicializado con 3
  elementos
```

$$\begin{bmatrix} 1 & x^2+y & \sqrt{2} \end{bmatrix} \quad \textbf{(1.1.3)}$$

```
> f:=(j)->x^(j-1): Vector[row](6,f); # vector fila inicializado cun
  procedemento ou función MAPLE
```

$$\begin{bmatrix} 1 & x & x^2 & x^3 & x^4 & x^5 \end{bmatrix} \quad \textbf{(1.1.4)}$$

```
> s:={1=0, 2=1}: Vector(2, s);
```

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (1.1.5)$$

```
> Vector[row](5, shape = constant[x])
```

$$\begin{bmatrix} x & x & x & x & x \end{bmatrix} \quad (1.1.6)$$

```
> Vector[row](3, shape = scalar[2, infinity])
```

$$\begin{bmatrix} 0 & \infty & 0 \end{bmatrix} \quad (1.1.7)$$

```
> Vector[row](5, shape = unit[3])
```

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (1.1.8)$$

Definición de matrices

Para construir unha matriz en MAPLE utilízase o comando

Matrix(r, c, init, ro, sym, sc, sh, st, o, dt, f, a)

onde os argumentos son:

r - (opcional) filas da matriz

c - (opcional) columnas da matriz

O resto dos valores son semellantes os do comando Vector. Os valores para *shape* son:

- *identity* : pon tódolos elementos a 0 e a diagonal a 1: *Matrix(3,5,shape= identity)*
- *scalar[x]*: pon tódolos elementos da diagonal a x, o resto a 0. Ex: *Matrix(3,3, shape=scalar[4.2])*
- *diagonal*: inicializa a diagonal co vector especificado en *init*. Ex: *Matrix(3,3, Vector([3,2,1]),shape=diagonal)* ou *f:=i->x^i:Matrix(4,f,shape=diagonal)*
- *triangular[upper/lower]*. Crea unha matriz triangular co valor especificado. Ex: *Matrix(3,3,5,shape=triangular[upper])*
- *symmetric* : Forza a que a matriz sexa simétrica: *Matrix(3,3,shape= symmetric); M[1,2]:=5* (fai tamén $M_{21}=5$)
- *antisymmetric* : Igual que *symmetric*, pero forza a que $M_{ij} = -M_{ji}$

Hai moitas outras máis, correspondentes con matrices especiais.

NOTA: Podemos convertir unha matriz en vector co comando *convert*:

```
> M := Matrix(2, 3, 5);
```

$$M := \begin{bmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix} \quad (1.2.1)$$

```
> convert(M, Vector[row])
```

$$\begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 \end{bmatrix} \quad (1.2.2)$$

```
> Matrix(2); Matrix(2,3); # por defecto, rechease con 0
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (1.2.3)$$

```
> Matrix(3, shape=identity); # matriz identidade
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1.2.4)$$

```
> m1:=Matrix(1..2, 1..3, 5); #definición da matriz especificando
  rango de índices (os rangos sempre empezan por 1) e recheada co
  número 5
```

$$m1 := \begin{bmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix} \quad (1.2.5)$$

```
> m2 := Matrix([[1, 2, 3], [4, 5, 6]]); #inicialización dunha
  matriz con datos de tipo list
```

$$m2 := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad (1.2.6)$$

```
> Matrix(m1+m2);
```

$$\begin{bmatrix} 6 & 7 & 8 \\ 9 & 10 & 11 \end{bmatrix} \quad (1.2.7)$$

```
> m3 := Matrix(3, 2, [1, 2, 3, 4, 5]); #outra inicialización da
  matriz con listas. O elemento (3,2) inicialízase por defecto a 0.
```

$$m3 := \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 0 \end{bmatrix} \quad (1.2.8)$$

```
> Matrix(4, 3, m3, fill = 9); # agora os elementos restantes
  inicialízanse por defecto a 9
```

$$\begin{bmatrix} 1 & 2 & 9 \\ 3 & 4 & 9 \\ 5 & 0 & 9 \\ 9 & 9 & 9 \end{bmatrix} \quad (1.2.9)$$

```
> f:=(i,j)->x^(i+j-1); Matrix(2,f); #inicialización cunha función
```

$$f := (i, j) \rightarrow x^{i+j-1}$$

(1.2.10)

$$\begin{bmatrix} x & x^2 \\ x^2 & x^3 \end{bmatrix} \quad (1.2.10)$$

```
> s:={(1,1)=0, (1,2)=1}; Matrix(1,3,s); # inicialización dos
  elementos da matriz utilizando índices
```

$$s := \{(1, 1) = 0, (1, 2) = 1\}$$

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \quad (1.2.11)$$

```
> Matrix(2, 3, symbol = m); # inicialización utilizando un símbolo
```

$$\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \end{bmatrix} \quad (1.2.12)$$

Tamén podemos crear unha matriz diagonal coa función **DIagonalMatrix** (require o módulo **LinearAlgebra**):

```
> v := Vector[row](3, [1, 2, 3]) : with(LinearAlgebra) : DiagonalMatrix(v)
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad (1.2.13)$$

Operacións con matrices

A transposición de matrices faise coa función Transpose con **with (LinearAlgebra)** incluido:

```
> with(LinearAlgebra):A:=Matrix(2,3,[1,2,3,4,5,6]);Transpose(A);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad (1.3.1)$$

A **suma** e a **resta** de matrices pode realizarse como a + e - de escalares.

```
> restart;
> A:=Matrix([[23,123,7],[22,17,18],[1,2,6]]);
```

$$A := \begin{bmatrix} 23 & 123 & 7 \\ 22 & 17 & 18 \\ 1 & 2 & 6 \end{bmatrix} \quad (1.3.2)$$

```
> B:=Matrix([[1, 10,5], [0,5,1], [1, 2,3]]);
```

(1.3.3)

$$B := \begin{bmatrix} 1 & 10 & 5 \\ 0 & 5 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad (1.3.3)$$

```
> A+B;
```

$$\begin{bmatrix} 24 & 133 & 12 \\ 22 & 22 & 19 \\ 2 & 4 & 9 \end{bmatrix} \quad (1.3.4)$$

```
> A-B;
```

$$\begin{bmatrix} 22 & 113 & 2 \\ 22 & 12 & 17 \\ 0 & 0 & 3 \end{bmatrix} \quad (1.3.5)$$

```
> A.B;
```

$$\begin{bmatrix} 30 & 859 & 259 \\ 40 & 341 & 181 \\ 7 & 32 & 25 \end{bmatrix} \quad (1.3.6)$$

```
> A^2; # cálculo da podencia dunha matriz
```

$$\begin{bmatrix} 3242 & 4934 & 2417 \\ 898 & 3031 & 568 \\ 73 & 169 & 79 \end{bmatrix} \quad (1.3.7)$$

```
> A^(-1); # cálculo da matriz inversa
```

$$\begin{bmatrix} -\frac{22}{4105} & \frac{724}{12315} & -\frac{419}{2463} \\ \frac{38}{4105} & -\frac{131}{12315} & \frac{52}{2463} \\ -\frac{9}{4105} & -\frac{77}{12315} & \frac{463}{2463} \end{bmatrix} \quad (1.3.8)$$

```
> 1/A;  # forma alternativa de calcular a matriz inversa
```

$$\begin{bmatrix} -\frac{22}{4105} & \frac{724}{12315} & -\frac{419}{2463} \\ \frac{38}{4105} & -\frac{131}{12315} & \frac{52}{2463} \\ -\frac{9}{4105} & -\frac{77}{12315} & \frac{463}{2463} \end{bmatrix} \quad (1.3.9)$$

Podemos borrar unha fila ou columna coa función **DeleteRow/DeleteColumn** na

libraría **LinearAlgebra**:

```
> with(LinearAlgebra) : DeleteRow(A, 2)
```

$$\begin{bmatrix} 23 & 123 & 7 \\ 1 & 2 & 6 \end{bmatrix} \quad \textbf{(1.3.10)}$$

Podemos ver as dimensións dunha matriz coa función **Dimension**:

```
> A; Dimension(A)
```

$$\begin{bmatrix} 23 & 123 & 7 \\ 22 & 17 & 18 \\ 1 & 2 & 6 \end{bmatrix}$$

$$3, 3 \quad \textbf{(1.3.11)}$$

Para ver a diagonal da matriz como vector columna, a función **Diagonal**:

```
> Diagonal(A)
```

$$\begin{bmatrix} 23 \\ 17 \\ 6 \end{bmatrix} \quad \textbf{(1.3.12)}$$

Para calcular a traza:

```
> Trace(A)
```

$$46 \quad \textbf{(1.3.13)}$$

A función **copy** serve para crear un duplicado de algo polo que nos permite realizar copias reais de vectores e matrices.

```
> C:=B; #non se realiza unha copia real senón que só se nomea a
  matriz B co nome C
```

$$C := \begin{bmatrix} 1 & 10 & 5 \\ 0 & 5 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad \textbf{(1.3.14)}$$

```
> C[1,1]:=70: B; # se modificamos o primeiro elemento de C e
  visualizamos B comprobamos que tamén se modificou
```

$$\begin{bmatrix} 70 & 10 & 5 \\ 0 & 5 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad \textbf{(1.3.15)}$$

```
> C1:=copy(B);
```

$$C1 := \begin{bmatrix} 70 & 10 & 5 \\ 0 & 5 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad \textbf{(1.3.16)}$$

```
> C1[1,1]:=a; C1;
```

$$C1_{1,1} := a$$

$$\begin{bmatrix} a & 10 & 5 \\ 0 & 5 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad (1.3.17)$$

```
> B; #B non se modificou, so C1
```

$$\begin{bmatrix} 70 & 10 & 5 \\ 0 & 5 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad (1.3.18)$$

Funcións básicas de álxebra linear

Neste apartado descríbense algunhas das funcións máis importantes da libraría **Linear Algebra**. Para poder utiliza-las funcións da libraría hai que primeiro cargalas na memoria do ordenador mediante o comando **with(libraría).**

```
> with(LinearAlgebra): #finalizando con : carga a libraría en
  memoria, se puxésemos ; cargaría a libraría en memoria e máis
  visualizaríamos tódalas funcións da libraría.
```

Determinante dunha matriz

A función ***Determinant(A, m)*** calcula o determinante da matriz A (m é un parámetro opcional que índica o método utilizado para calcula-lo determinante). A matriz pode estar definida de xeito numérico ou simbólico.

```
> B:=Matrix([[2,23,1],[2,4,6],[1,7,3]]);
```

$$B := \begin{bmatrix} 2 & 23 & 1 \\ 2 & 4 & 6 \\ 1 & 7 & 3 \end{bmatrix} \quad (1.4.1.1)$$

```
> d:=Determinant(B);
```

$$d := -50 \quad (1.4.1.2)$$

Matriz característica e polinomio característico

A función **CharacteristicMatrix(A, lambda, outopts)** constrúe a matriz característica da matriz A (é dicir, lambda*I-A, sendo I a matriz identidade), onde **A** é unha matriz cadrada e **lambda** a variabel que se usa.

```
> A := Matrix([[1, 2, 3], [1, 2, 3],[1, 5, 6]]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 5 & 6 \end{bmatrix} \quad (1.4.2.1)$$

```
> CharacteristicMatrix(A, lambda);
```

(1.4.2.2)

$$\begin{bmatrix} \lambda - 1 & -2 & -3 \\ -1 & \lambda - 2 & -3 \\ -1 & -5 & \lambda - 6 \end{bmatrix} \quad \textbf{(1.4.2.2)}$$

A función **CharacteristicPolynomial(A, lambda)** cálcula o polinomio característico da matriz cadrada A (é dicir, **(-1)^n * det(A-lambda*I**), onde I é a matriz identidade e n a dimensión da matriz A).

```
> CharacteristicPolynomial(A, lambda);
```

$$\lambda^3 - 9\lambda^2 \quad \textbf{(1.4.2.3)}$$

Cálculo dos valores e vectores propios dunha matriz cadrada

A función **Eigenvalues** cálcula os valores propios e a función **Eigenvectors** cálcula os vectores propios dunha matriz cadrada. A sintase das funcións é: **Eigenvalues(A, C, imp, o, outopts)** e **Eigenvectors(A, C, imp, o, outopts)** onde **A** é a matriz a calcular os valores propios. Os demáis argumentos son opcionais e representan: **C** e a matriz para o caso xeneralizado, **imp** é unha variabel lóxica que nos dí se vai devolve-los valores como raíz dunha ecuación(RootOf) ou como radicais, **o** é o obxeto no que queremos que se devolva o resultado (Vector, Vector [row], Vector[column] ou list) e **outopts** fai referencias as opcións de construcción do obxeto de saída.

```
> R := Matrix([[611, 196, -192, 407, -8, -52, -49, 29], [899, 113,
  -192, -71, -43, -8, -44], [899, 196, 61, 49, 8, 52], [611, 8, 44,
  59, -23], [411, -599, 208, 208], [411, 208, 208], [99, -911],
  [99]], shape = symmetric, scan = triangular[upper]);
```

$$R := \begin{bmatrix} 611 & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\ 196 & 899 & 113 & -192 & -71 & -43 & -8 & -44 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 52 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ -52 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ -49 & -8 & 8 & 59 & 208 & 208 & 99 & -911 \\ 29 & -44 & 52 & -23 & 208 & 208 & -911 & 99 \end{bmatrix} \quad \textbf{(1.4.3.1)}$$

```
> Eigenvalues(R);
```

(1.4.3.2)

$$\begin{bmatrix} 0 \\ 1020 \\ 510+100\sqrt{26} \\ 510-100\sqrt{26} \\ 10\sqrt{10405} \\ -10\sqrt{10405} \\ 1000 \\ 1000 \end{bmatrix} \quad \textbf{(1.4.3.2)}$$

```
> A := Matrix([[-1, -3, -6],[3, 5, 6], [-3, -3, -4]]);
```

$$A := \begin{bmatrix} -1 & -3 & -6 \\ 3 & 5 & 6 \\ -3 & -3 & -4 \end{bmatrix} \quad \textbf{(1.4.3.3)}$$

```
> v, e := Eigenvectors(A); #o vector v son os valores propios e a
  matriz e son os vectores propios
```

$$v, e := \begin{bmatrix} 2 \\ 2 \\ -4 \end{bmatrix}, \begin{bmatrix} -2 & -1 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix} \quad \textbf{(1.4.3.4)}$$

Triangulación dunha matriz

A función **GaussianElimination(A, m, outopts)** realiza a triangulación dunha matriz cadradautilizando eliminación gausiana. O resultado é unha matriz triangular superior das mesmas dimensións que a matriz A. Os argumentos son: **A** a matriz a triangulizar, os outros dous argumentos **m** e **outopts** son opcionais e especifican o método utilizado e outras opcións avanzadas.

```
> A :=Matrix([[8, 3, -1, -5], [4, -5, 0, -2], [-5, 8, 3, -1],[-5,
  5, -4, -9]]);
```

$$A := \begin{bmatrix} 8 & 3 & -1 & -5 \\ 4 & -5 & 0 & -2 \\ -5 & 8 & 3 & -1 \\ -5 & 5 & -4 & -9 \end{bmatrix} \quad \textbf{(1.4.4.1)}$$

```
> GaussianElimination(A);
```

(1.4.4.2)

$$\begin{bmatrix} 8 & 3 & -1 & -5 \\ 0 & -\frac{13}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{163}{52} & -\frac{175}{52} \\ 0 & 0 & 0 & -\frac{2607}{163} \end{bmatrix} \quad \textbf{(1.4.4.2)}$$

Resolución dun sistema de ecuacións lineares

A función **LinearSolve(A, B, m, t, c, ip, outopts, methopts)** resolve un sistema lineal de ecuacións do tipo **Ax=B** (devolve o vector x que satisface o sistema). O argumento **A** e unha matriz ou lista. Os demáis argumentos son opcionais: **B** é unha matriz ou vector columna, **m** especifica o método de resolución, **t** especifica as variabeis libres en solucións parametrizadas e as demáis son opcións avanzadas.

```
> M := <<(1, 1, 1, 4)>|<(1, 1, -2, 1)>|<(3, 1, 1, 8)>|<(-1, 1, -1,
  -1)>|<(0, 1, 1, 0)>>;
```

$$M := \begin{bmatrix} 1 & 1 & 3 & -1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & -2 & 1 & -1 & 1 \\ 4 & 1 & 8 & -1 & 0 \end{bmatrix} \quad \textbf{(1.4.5.1)}$$

```
> LinearSolve(M);
```

$$\begin{bmatrix} \frac{25}{6} \\ \frac{4}{3} \\ -\frac{5}{2} \\ -2 \end{bmatrix} \quad \textbf{(1.4.5.2)}$$