Curso basico de MATLAB

8. Vectores y matrices

8.1 Vectores

8.1.1 Posiciones de un vector

Los elementos de un vector se identifican por el índice: El índice representa cada posición del vector. Se puede hacer referencia a datos de un vector a través de los índices guardados en una variable.
Siendo:
» w = [1 6 2 8 12 13];

- Se puede hacer referencia a un valor o una posición del vector **w**, simplemente indicando entre paréntesis la posición que ocupa dicho valor dentro del vector.
  Por ejemplo si se quisiera trabajar con el valor 8, que ocupa la cuarta posición del vector **w** se le indica al ***Matlab*** así:
  »w(4)
  ans =
  8

- Se puede modificar un elemento del vector simplemente cambiando su valor haciendo referencia a la posición del valor a cambiar así:
  » w(3) =15
  w =
  1 6 15 8 12 13

- Si se asigna un valor en una posición que no existe y está adelante de la última posición el ***Matlab*** asigna ceros a las posiciones restantes.
  » w(10) = 7
  w =
  1 6 15 8 12 13 0 0 0 *7* *Asignó ceros a las posiciones 7,8 y 9.*

- Un índice puede ser un vector. Si **x** y **v** son vectores, entonces
  x(v) = [x(v(1)), x(v(2)), ...,x(v(n))]. El vector ***v*** es para direccionar al vector **x**.

  » x = [8 7 9 5 6];
  » v = [2 4 1];
  » t = x(v)
  t =
  7 5 8 **t** contiene los elementos de la segunda, la cuarta y la primer posición del vector **x**

8.1.2 Operador dos puntos (:) para vectores

Se podría decir que este operador representa un *rango.* Se puede hacer referencia a un rango de valores del vector indicando las posiciones de dicho rango.

Siendo:
» w = [1 6 2 8 12 13];

- Por ejemplo si se quisiera referenciar los valores 6, 2 y 8 simplemente se hace referencia desde la segunda posición hasta la cuarta así:
  » w(2:4)
  ans =
  6 2 8

- Por ejemplo si se quisiera referenciar los valores 6, 8 y 13 simplemente se hace referencia de las posiciones pares desde la segunda hasta la sexta así:
  » w(2:2:6)
  ans =
  6 8 13

8.1.3 Ejemplos de utilización del operador dos puntos (:) para crear vectores

- Se asignan a la variable **x** los valores del 1 al 5 de forma secuencial de 1 en 1
  » x = 1:5
  x =
  1 2 3 4 5

Por defecto el incremento es 1, pero este operador puede también utilizarse con otros valores enteros y reales, positivos o negativos.

- Se asignan a la variable **x**, los valores del 5 al 1 de forma secuencial inversa de 1 en 1
  » x = 5:-1:1
  x =
  5 4 3 2 1

- Se asignan a la variable **x**, los valores del 0 al 1 con incrementos de 0.25
  » x = 0:0.25:1
  x =
  0 0.2500 0.5000 0.7500 1.0000

- Se asignan a la variable **x**, los valores del 0 al 10 con incrementos de un valor **m**
  » m = 3.12;
  » x = 0:m:10
  x =
  0 3.12 6.24 9.36

8.1.4 Unión de vectores

- Se pueden unir vectores de la siguiente manera

  Siendo:
  » x = 1:5;
  » y = sin(x);

  Hacemos:
  » [x y]
  ans =
  1 2 3 4 5 0.84147 0.9093 0.14112 -0.7568 -0.95892

  El resultado es un vector de diez elementos en donde los primeros cinco elementos corresponden al vector **x** y los últimos cinco elementos corresponden al vector **y**.

- Se pueden generar vectores con rangos de valores de otros vectores.

  Siendo:
  » x = 1:100;
  » y = sin(x);

  Hacemos:
  » z = [x(1:10),y(1:5)]
  z =
  1 2 3 4 5 6 7 8 9 10 0.84147 0.9093 0.14112 -0.7568 -0.95892

8.1.5 Transposición de vectores

Si se trata de asignar un rango de valores en forma de columna, se puede asignar el rango inicialmente a un vector fila y luego transponerlo en un vector columna o al contrario, si se tiene un vector columna se puede asignar dichos valores a un vector fila.

» a = [2;1;2]
a =
2
1
2

Se transpone a un vector fila:
» a'
ans = 2 1 2

Otra forma de transponer un vector es la siguiente:
» a = 1:10
a =
1 2 3 4 5 6 7 8 9 10

```
» b = a(:)
b =
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```

8.1.6 El comando LENGTH

Determina el número de componentes o elementos de un vector.
La sintaxis es:

Longitud = length (vector);

```
» x = [1 2 3 4 5 6 7 ];
» l = length(x)
l =
     7
```

8.1.7 El comando FIND

Encuentra los índices del vector donde se cumple una determinada condición.
La sintaxis es:

Variable = find (condición);

find(x)	Obtiene los índices del vector x de los elemento distintos de cero.
find(x>3)	Obtiene los índices del vector x de cada elemento > 3

Por ejemplo: generar un vector **B** cuyos elementos sean los índices del vector **A** cuyos elementos sean números pares

```
» A = [2 4 5 7 4];
» B = find (A/2 = = fix (A/2) )
B = [ 1 2 5 ]
```

8.1.8 Algunas funciones que actúan sobre vectores

max(x)	Devuelve el mayor elemento del vector x
[Xmax,imax]=max(x)	Asigna el mayor elemento de x a la variable ***Xmax*** y la posición que ocupa en x a la variable ***imax***
min(x)	Devuelve el menor elemento del vector x.
[Xmin,imin]=min(x)	Asigna el menor elemento de x a la variable ***Xmin*** y la posición que ocupa en x a la variable ***imin***
sum(x)	Devuelve la suma de los elementos del vector x.
cumsum(x)	Devuelve un vector con la suma acumulativa de los elementos del vector x.
prod(x)	Devuelve le producto de los elementos de un vector
cumprod(x)	Devuelve un vector con el producto acumulativo de los elementos del vector x.
mean(x)	Devuelve el promedio de los elementos del vector x.
median(x)	Devuelve el valor medio de los elementos del vector x.
sort(x)	Devuelve un vector con los elementos del vector x ordenados en forma ascendente.
[z,i]=sort(x)	Devuelve un vector ***z*** con los elementos de x ordenados en forma ascendente y un vector ***i*** con las posiciones iniciales en x de los elementos en el vector ordenado.
dsort(x)	Devuelve un vector con los elementos del vector x ordenados en forma descendente.
std(x)	Devuelve la desviación estándar.
norm(x)	Devuelve la norma del vector x.

Algunas de estas funciones *se pueden aplicar también a matrices*

8.1.9 Algunas formas para generar vectores

linspace(a, b)	Genera un vector de 100 valores igualmente espaciados entre a y b.
linspace(a, b, N)	Genera un vector de N valores igualmente espaciados entre a y b.
logspace(a, b, N)	Genera un vector de N valores logaritmicamente espaciados entre a y b.
zeros(n,1)	Genera un vector columna de n ceros.
zeros(1,n)	Genera un vector fila de n ceros.
ones(n,1)	Genera un vector columna de n unos.
ones(1,n)	Genera un vector fila de n unos.
rand(n,1)	Genera un vector columna de n números aleatorios entre 0 y 1, con distribución uniforme.
rand(1,n)	Genera un vector fila de n números aleatorios entre 0 y 1, con distribución uniforme.
randn(n,1)	Genera un vector columna de n números aleatorios con distribución normal, de valor medio 0 y varianza 1.
randn(1,n)	Genera un vector fila de n números aleatorios con distribución normal, de valor medio 0 y varianza 1.

8.2 Matrices

Para definir una matriz *no hace falta establecer de antemano su tamaño* (de hecho, se puede definir un tamaño y cambiarlo posteriormente). ***Matlab*** determina el número de filas y de columnas en función del número de elementos que se proporcionan. Los elementos de la matriz pueden ser números o expresiones.

8.2.1 Posiciones de una matriz

Las matrices se definen por filas y columnas. Existe un índice para referenciar la fila y otro para la columna.
Para ingresar una matriz en ***Matlab,*** los elementos de una misma fila están separados por ***espacios blancos*** o ***comas***, mientras que las filas están separadas por caracteres ***punto y coma*** (;). También es posible, ingresar fila por fila separados por un ***enter*** entre ellas.

Siendo:

```
» A =
    35   1   6  26  19  24
     3  32   7  21  23  25
    31   9   2  22  27  20
     8  28  33  17  10  15
    30   5  34  12  14  16
     4  36  29  13  18  11
```

Si se quiere hacer referencia al elemento de la segunda fila y tercera columna:

```
» A(2,3)
ans =
     7
```

Para extraer los elementos de una matriz A que se encuentran en la intersección de las filas n-ésima y m-ésima y las columnas p-ésima y q-ésima

```
» B = A([m, n],[p, q]);
```

Siendo:

```
» A = [1 2 3 4; 4 5 6 7; 7 8 9 10; 8 9 10 11]
A =
    1  2  3  4
    4  5  6  7
    7  8  9 10
    8  9 10 11
```

Hacemos:

```
» B = A([1 3],[2 4])
B =
    2   4
    8  10
```

8.2.2 Matriz nula o vacía

Para ***Matlab*** una matriz definida sin ningún elemento entre los corchetes es una matriz que existe, pero que está vacía, o lo que es lo mismo que tiene dimensión cero. Por ejemplo:

```
» B=[ ]
B =
        [ ]

» exist(B)
ans =
        [ ]

» isempty(B)
ans =
        1
```

Con el comando ***exist( )***, constatamos que la matriz existe y con el comando ***isempty( )*** verificamos que esta vacía.

8.2.3 Operador dos puntos (:) para matrices (manipulación de los elementos)

Al igual que con los vectores se podría decir que este operador representa un *rango.* Los dos puntos aislados representan "todos los elementos". Se utilizan para extraer una submatriz a partir de una matriz.

- Para generar un vector cuyos elementos son las todas las columnas de una matriz A situadas por orden
  » B = A(:)

- Para extraer la primera fila de una matriz A
  » B = A(1,:);

- Para acceder a la última fila o columna puede utilizarse la palabra ***end***, en lugar del número correspondiente.
  » A(end, :);

- Para extraer la segunda columna de una matriz A
  » B = A(:,2);

- Para extraer los cuatro primeros elementos de la sexta fila de una matriz A
  » B = A(6, 1:4)

- Para extraer las filas desde la primera hasta la tercera y las columnas desde la primera hasta la cuarta de una matriz A
  » B = A(1:3,1:4);

- Para extraer conjuntos disjuntos de filas utilizando corchetes [ ]. Por ejemplo, extraer la primera, segunda y quinta fila de una matriz A.
  » A([1 2 5],:);

- Para eliminar todos los elementos de la tercer columna de una matriz A.
  » A(:,3)=[ ]

- Para extraer las filas que hay entre la a-ésima y la b-ésima posición tomándolas de p en p, y las columnas que hay entre la c-ésima y la d-ésima tomándolas de q en q.
  » A(a:p:b,c:q:d)

8.2.4 Unión de matrices y redimensionamiento

- Para unir matrices hacemos:

```
» A=[1 2;3 4]
A =
    1    2
    3    4

» B=[5 6;7 8]
B =
    5    6
    7    8

» C=[A;B]
C =
    1    2
    3    4
    5    6
    7    8

  » C=[A B]
  C =
      1    2    5    6
      3    4    7    8
```

- Para redimensionar matrices: *usamos el comando* ***reshape***

Los elementos de la matriz original son colocados en el número de filas y columnas descrito para la matriz modificada. El orden de colocación es de arriba hacia abajo y de izquierda a derecha. El numero de elementos debe ser el mismo en las dos matrices:
Sintaxis:

Matriz_modificada = reshape(Matriz_original, filas, columnas)

Ejemplo:

```
» A = [1 4 7 10; 2 5 8 11; 3 6 9 12]
A =
        1 4 7 10
        2 5 8 11
        3 6 9 12

» B = reshape(A,2,6)
B =
        1  3  5  7   9 11
        2  4  6  8 10 12
```

Notar que el orden de la matriz A es de 3x4 es decir 12 elementos que debe ser igual al número de elementos del nuevo orden 2x6 de B.

8.2.5 Algunas funciones que actúan sobre matrices

poly(A)	Devuelve un vector con los coeficientes del polinomio característico de la matriz cuadrada A
trace(A)	Devuelve la traza (suma de los elementos de la diagonal) de una matriz cuadrada A
size(A)	Devuelve el tamaño de una matriz cuadrada A
[m,n] = size(A)	Devuelve el número de filas m y de columnas n de una matriz rectangular A.
norm(A)	Devuelve la norma de una matriz A.
rank(A)	Devuelve el rango de una matriz A.
det(A)	Devuelve el determinante de una matriz cuadrada A.
flipud(A)	Devuelve la matriz simétrica de A respecto de un eje horizontal
fliplr(A)	Devuelve la matriz simétrica de A respecto de un eje vertical
triu(A)	Devuelve la matriz triangular superior a partir de una matriz A (no tiene porqué ser cuadrada)
tril(A)	Devuelve la matriz triangular inferior a partir de una matriz A (no tiene porqué ser cuadrada).
reshape(A,m,n)	Cambia el tamaño de la matriz A devolviendo una matriz de tamaño m x n cuyas columnas se obtienen a partir de un vector formado por las columnas de A puestas una a continuación de otra. Si la matriz A tiene menos de m x n elementos se produce un error.

8.2.6 Algunas formas para generar matrices

zeros(n)	Genera una matriz de n x n ceros.
zeros(n,m)	Genera una matriz de n x m ceros.
ones(n)	Genera una matriz de n x n unos.
ones(n,m)	Genera una matriz de n x m unos.
eye(n)	Genera una matriz unidad de orden n x n.
rand(n)	Genera una matriz de n x n números aleatorios entre 0 y 1, con distribución uniforme.

rand(n,m)	Genera una matriz de n x m números aleatorios entre 0 y 1, con distribución uniforme.
randn(n)	Genera una matriz de n x n números aleatorios con distribución normal, de valor medio 0 y varianza 1.
randn(n,m)	Genera una matriz de n x m números aleatorios con distribución normal, de valor medio 0 y varianza 1.
magic(n)	Genera una matriz de n x n con la propiedad de que todas las filas y columnas suman lo mismo.
A=diag(x)	Genera una matriz diagonal A cuyos elementos diagonales son los elementos de un vector ya existente x

8.3 Otras funciones que actúan sobre vectores y matrices

isempty(x)	Chequea si un vector x está vacío
any(x)	Determina si el vector contiene algún valor no cero
isnan(A)	Chequea si hay valores *NaN* en la matriz A, devolviendo una matriz de unos y ceros del mismo tamaño que A.
isinf(A)	Chequea si hay valores *Inf* en la matriz A, devolviendo una matriz de unos y ceros del mismo tamaño que A.
isfinite(A)	Chequea si los valores de la matriz A son finitos.