2. Variables numéricas: Vectores y matrices en Matlab

- Variables
- Creación de vectores y matrices.
- Dimensión de una matriz
- Concatenar matrices o vectores
- Matriz traspuesta
- Matriz inversa
- Determinante de una matriz
- Selección y/o modificación de elementos o partes de un vector o matriz
- Operaciones con vectores y matrices.
    - Suma
    - Producto
    - Producto de matrices
    - Producto de matriz por vector
    - Producto escalar de dos vectores
    - Operaciones de una matriz con un escalar
    - Operaciones con la matriz inversa

Variables

En un lenguaje de programación una variable es un espacio en la memoria del ordenador al que se le ha asignado un nombre; el contenido de dicho espacio puede cambiar a lo largo de una sesión de trabajo o durante la ejecución de un programa (precisamente por eso se denomina *variable*).

Por ejemplo, la ejecución del comando:

```
>> a = 25
```

```
a =  25
```

crea un espacio llamado `a` en la memoria, en el que se guarda el valor 25. Decimos entonces que hemos creado la variable `a`, cuyo valor inicial es 25. Si ahora hacemos:

```
>> a = 30
```

```
a =  30
```

habremos cambiado el valor de `a` del anterior 25 al nuevo valor 30. Como iremos viendo a lo largo de esta asignatura, las variables pueden ser de distintos tipos (numéricas, de texto, lógicas, de fecha, …). Asimismo el contenido de una variable puede ser un único valor como en este ejemplo, o puede ser un vector, una matriz, o una estructura más compleja.

Creación de vectores y matrices.

En Matlab un vector se crea simplemente escribiendo los valores que lo componen entre corchetes. La siguiente linea crea el vector $(1, 2, 3)$ y lo asigna a la variable `V`:

⊕

```
>> V=[1 2 3]
```

```
V =

   1   2   3
```

Los componentes de un vector se pueden declarar, tal como acabamos de hacer, simplemente separándolos por espacios. Alternativamente, en lugar de espacios podemos

separar los componentes del vector por comas; el resultado es el mismo:

```
>> V=[1, 2, 3]
```

```
V =


   1   2   3
```

Una matriz es igualmente fácil de crear: puede definirse fila por fila en lineas separadas:

⊕

```
>> A=[1 2 3 4 5
    6 7 8 9 10
    11 12 13 14 15]
```

```
A =


    1    2    3    4    5
    6    7    8    9   10
   11   12   13   14   15
```

O también pueden definirse las distintas filas en una sola linea, si bien en tal caso hay que separarlas con punto y coma:

```
>> A=[1 2 3 4 5;  6 7 8 9 10;  11 12 13 14 15]
```

```
A =


    1    2    3    4    5
    6    7    8    9   10
   11   12   13   14   15
```

Por defecto, cada vez que declaramos un vector o una matriz, Matlab nos muestra inmediatamente a continuación la asignación realizada. Si queremos evitar este comportamiento, bastará con terminar la declaración del vector (o matriz) con un punto y coma:

```
>> V=[1 2 3];
```

```
>> A=[1 2 3 4 5
    6 7 8 9 10
    11 12 13 14 15];
```

En general para Matlab los vectores son también matrices con una única fila (o una única columna).

Dimensión de una matriz

⊕ La función `size` nos devuelve el tamaño (número de filas y columnas) de una matriz;

```
>> size(A)
```

```
ans =


   3   5
```

Si especificamos `size(A,1)` obtenemos el **número de filas** de A:

```
>> size(A,1)
```

```
ans =  3
```

Asimismo, `size(A,2)` nos devuelve el **número de columnas** de A:

```
>> size(A,2)
```

```
ans =  5
```

⊕ La función `numel` devuelve el número total de elementos de una matriz:

```
>> numel(A)
```

```
ans =  15
```

⊕ Por último, la función `length` aplicada a un vector nos da su dimensión (número de elementos que lo componen); aplicada a una matriz, nos devuelve su número de columnas:

```
>> length(V)
```

```
ans =  3
```

```
>> length(A)
```

```
ans =  5
```

Concatenar matrices o vectores

Octave es capaz de concatenar o “pegar” matrices o vectores en una nueva matriz o vector siempre que las dimensiones originales encajen adecuadamente. Por ejemplo si hacemos:

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

```
A =

   1   2
   3   4
```

```
>> B= [8; 7]
```

```
B =

   8
   7
```

⊕

- “Pegamos” B a la derecha de A:

```
>> C=[A B]
```

```
C =

   1   2   8
   3   4   7
```

- Añadimos una nueva fila a la matriz anterior:

```
>> D=[2 3 5]
```

```
D =
```

```
   2    3    5
```

```
>> [A B; D]
```

```
ans =


   1    2    8
   3    4    7
  -2    3   -5
```

- “Pegamos” el vector V a la derecha de V:

```
>> [V V]
```

```
ans =


   1    2    3    1    2    3
```

- “Pegamos” el vector V debajo de V, creando una matriz de dimensión 2×3:

```
>> [V; V]
```

```
ans =


   1    2    3
   1    2    3
```

Matriz traspuesta

La traspuesta de una matriz (o vector) se obtiene simplemente añadiendo un apóstrofe (’) al nombre de la matriz o vector:

```
>> A=[1 2 3;  4 5 6;  7 8 9];
```

⊕

```
>> A'
```

```
ans =


   1    4    7
   2    5    8
   3    6    9
```

```
>> V=[1 2 3];
```

⊕

```
>> V'
```

```
ans =

   1
   2
   3
```

Matriz inversa

La inversa de una matriz se obtiene mediante la función `inv()`:

```
>> A=[1 1; 1 2];
```

⊕

```
>> inv(A)
```

```
ans =

   2  -1
  -1   1
```

En caso de que la matriz no tenga inversa, Matlab nos muestra un aviso (*warning*):

```
>> B=[1 1; 2 2];
```

```
>> inv(B)
```

```
warning: matrix singular to machine precision
ans =

   Inf   Inf
   Inf   Inf
```

Determinante de una matriz

El determinante de una matriz se obtiene mediante la función `det()`:

```
>> A=[1 -3; 2 5];
```

⊕

```
>> det(A)
```

```
ans =  11
```

Selección y/o modificación de elementos o partes de un vector o matriz

Una vez que hayamos definido un vector o matriz en Matlab podemos acceder a sus elementos especificando entre paréntesis, a continuación del nombre del vector o matriz, la posición (o posiciones) a que queremos acceder. Si por ejemplo, definimos la matriz A siguiente:

```
>> A=[4 6 8 7 9; 1 -2 -7 8 4; 2 3 5 5 7]
```

```
A =

   4   6   8   7   9
   1  -2  -7   8   4
   2   3   5   5   7
```

podemos seleccionar el término $A_{2,4}$ que ocupa la posición (2,4) (fila 2, columna 4) mediante:

⊕

```
>> A(2,4)
```

```
ans =  8
```

También podemos seleccionar fácilmente una submatriz, indicando las filas y columnas iniciales y finales:

⊕

```
>> A(2:3,3:4)
```

```
ans =

  -7   8
   5   5
```

Podemos hacer selecciones más complicadas; por ejemplo, si queremos seleccionar la submatriz formada por las filas 1 y 3 y las columnas 2 y 5:

```
>> A([1 3],[2 5])
```

```
ans =

   6   9
   3   7
```

Para seleccionar una fila completa especificamos ":" como índice de columna. Por ejemplo, para mostrar las dos primeras filas completas:

⊕

```
>> A([1 2],:)
```

```
ans =

   4   6   8   7   9
   1  -2  -7   8   4
```

Asimismo, para seleccionar una columna completa especificamos ":" como índice de fila. Por ejemplo, para mostrar sólo la cuarta columna:

⊕

```
>> A(:,4)
```

```
ans =

   7
   8
   5
```

Podemos utilizar las selecciones anteriores para modificar los valores correspondientes en la matriz; por ejemplo, si queremos cambiar **todos** los términos de la cuarta columna por unos:

```
>> A(:,4)=1
```

```
A =

   4   6   8   1   9
   1  -2  -7   1   4
   2   3   5   1   7
```

Para cambiar los términos de la segunda fila por los números 5,4,3,2,1 en ese orden:

```
>> A(2,:)=[5 4 3 2 1]
```

```
A =

   4   6   8   1   9
   5   4   3   2   1
   2   3   5   1   7
```

Para borrar una fila (o columna) de una matriz la cambiamos por la **matriz vacía** `[]`. Por ejemplo, para borrar la tercera columna de la matriz anterior:

```
>> A(:,3)=[]
```

```
A =

   4   6   1   9
   5   4   2   1
   2   3   1   7
```

Y si ahora queremos borrar las filas 1 y 3:

```
>> A([1 3],:)=[]
```

```
A =

   5   4   2   1
```

Por último, si se asignan valores a partes de matrices o vectores que no se han definido

previamente, Matlab rellena el resto de la matriz
o vector con ceros:

```
>> d([1 3 5 7])=-1
```

```
d =


  -1   0  -1   0  -1   0  -1
```

```
>> H(3,4)=5
```

```
H =


   0   0   0   0
   0   0   0   0
   0   0   0   5
```

Operaciones con vectores y matrices.

Suma

```
>> V=[1 2 3];
```

```
>> U=[-1 3 4];
```

⊕

```
>> U+V
```

```
ans =


   0   5   7
```

⊕

```
>> A=[1 2 3; 4 5 6; 7 8 9]
```

```
A =


   1   2   3
   4   5   6
   7   8   9
```

```
>> B=[0 1 -2; 3 2 -4; 1 1 1]
```

```
B =

   0   1  -2
   3   2  -4
   1   1   1
```

```
>> A+B
```

```
ans =

    1    3    1
    7    7    2
    8    9   10
```

Producto

Producto de matrices

⊕

```
>> A*B
```

```
ans =

    9    8   -7
   21   20  -22
   33   32  -37
```

Producto de matriz por vector

⊕

```
>> A=[1 2 3; 4 5 6; 7 8 9];
 V=[1 2 3];
 A*V'
```

```
ans =

   14
   32
   50
```

Producto escalar de dos vectores

⊕

```
>> U*V'
```

```
ans =  17
```

Operaciones de una matriz con un escalar

Cuando se desea multiplicar una matriz o un vector por un escalar se utiliza el símbolo ".*" (punto seguido de asterisco). Así por ejemplo, para multiplicar por 2 todos los valores de la matriz *A* anterior:

```
>> 2.*A
```

```
ans =

    2    4    6
    8   10   12
   14   16   18
```

Si queremos sumar la misma cantidad a todos los términos de una matriz usamos “.+”. Para sumar 3 a todos los términos de la matriz *A* anterior:

```
>> 2.+A
```

```
ans =

    3    4    5
    6    7    8
    9   10   11
```

```
>> 3.*V
```

```
ans =

   3   6   9
```

Operaciones con la matriz inversa

Supongamos ahora que tenemos las siguientes matrices:

```
>> A=[1 3 0; 2 1 -3; 0 1 2];
 B=[2, 1, 4; 6 0 1; 1 0 0];
```

Si queremos multiplicar A por la inversa de B podemos ejecutar:

```
>> A*inv(B)
```

```
ans =

    3  -12   67
    1   -7   42
    1   -2   10
```

Nótese que, de alguna manera, multipicar A por la inversa de B viene a ser como dividir A por B. Matlab nos permite ejecutar esta operación como:

```
>> A/B
```

```
ans =

    3  -12   67
    1   -7   42
    1   -2   10
```

y obtenemos el mismo resultado que antes.

Si quisiéramos realizar el producto $B^{-1} \cdot A$, podemos calcularlo como:

```
>> inv(B)*A
```

```
ans =

    0    1    2
   -7   21   56
    2   -5  -15
```

pero matlab nos permite también utilizar, de manera equivalente, la siguiente notación algo más simple:

```
>> B\A
```

```
ans =


    0     1     2
   -7    21    56
    2    -5   -15
```

Nótese que podemos leer el símbolo “\”, en la operación `B\A` como que la primera matriz “divide” a la segunda.