PRÁCTICA 1

Vectores y MATLAB

Esta y todas las demás prácticas están pensadas para ser trabajadas delante de un ordenador con MATLAB instalado, y no para ser leídas como una novela. En vez de eso, cada vez que se presente un comando de MATLAB, se debe introducir el comando, pulsar la tecla "Enter" para ejecutarlo y ver el resultado. Más aún, se desea que se verifique el resultado. Asegúrese de que se comprende perfectamente lo que se obtiene antes de continuar con la lectura.

Aunque MATLAB es un entorno que trabaja con matrices, en esta práctica se aprenderá cómo introducir vectores por filas o por columnas y a manejar algunas operaciones con vectores.

Prerrequisitos: ninguno.

1. Vectores fila

La introducción de vectores fila en MATLAB es muy fácil. Introdúzcase el siguiente comando en la pantalla de MATLAB [1]

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

Hay una serie de ideas a destacar en este comando. Para introducir un vector, se escribe una apertura de corchete, los elementos del vector separados por espacios y un cierre de corchete. Se pueden usar también comas para delimitar las componentes del vector

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

El signo = es el operador de asignación de MATLAB. Se usa este operador para asignar valores a variables. Para comprobar que el vector fila [1,2,3] ha sido asignado a la variable v introdúzcase el siguiente comando en el indicador de MATLAB.

[1] El símbolo >> es el indicador de MATLAB. Se debe introducir lo que aparece tras el indicador. Entonces se pulsa la tecla "Enter" para ejecutar el comando.

```
>> v
```

1.1. Rangos.

Algunas veces es necesario introducir un vector con componentes a intervalos regulares. Esto se realiza fácilmente con `MATLAB` con la estructura `inicio:incremento:fin`. Si no se proporciona un incremento, `MATLAB` asume que es 1.

```
>> x1=0:10
```

Se puede seleccionar el propio incremento.

```
>> x2=0:2:10
```

Se puede ir incluso hacia atrás.

```
>> x3=10:-2:1
```

O se le puede echar imaginación.

```
>> x4=0:pi/2:2*pi
```

Hay veces, sobre todo cuando hay que pintar funciones, que se precisan un gran número de componentes en un vector.

```
>> x=0:.1:10
```

1.2. Elimina la salida.

Se puede suprimir la salida de un comando de `MATLAB` añadiendo un punto y coma.

```
>> x=0:.1:10;
```

Es muy útil cuando la salida es muy grande y no se desea verla.

1.3. Espacio de trabajo de MATLAB.

Es posible obtener una lista de las variables en el espacio de trabajo en cualquier momento mediante el comando

```
>> who
```

Se puede obtener incluso más información acerca de las variables con

```
>> whos
```

Se eliminar la asignación hecha a una variable con

```
>> clear x
>> who
```

Obsérvese que también se da el tamaño de cada variable. Es posible mantener una ventana con la lista de variables usadas y su tamaño. Para ello, en la barra superior selecciónese el menú `Desktop` y actívese la opción `Workspace`.

Se puede obtener el tamaño de un vector `v` con el comando

```
>> size(v)
```

La información que devuelve indica que el vector `v` tiene 1 fila y 3 columnas. Aunque se puede entender al vector `v` como una matriz con 1 fila y 3 columnas, también se puede entender como un vector fila de longitud 3. Por ejemplo, pruébese el siguiente comando:

```
>> length(v)
```

2. Vectores columna

Es también fácil escribir vectores columna en MATLAB. Introdúzcase el siguiente comando en el indicador.

```
>> w=[4;5;6]
```

Observe que los símbolos de punto y coma delimitan las filas de un vector columna. Pruébense los siguientes comandos.

```
>> w
>> who
>> whos
>> size(w)
```

El resultado indica que el vector `w` tiene 3 filas y 1 columna. Aunque se puede ver al vector `w` como una matriz de 3 filas y 1 columna, también es posible pensar en él como un vector columna de longitud 3. Pruébese el siguiente comando.

```
>> length(w)
```

2.1. Transposición.

El operador en `MATLAB` para transponer es el apóstrofe simple `'`. Se puede cambiar así un vector fila a un vector columna.

```
>> y=(1:10)'
```

O un vector columna a un vector fila.

```
>> y=y'
```

2.2. Indexado de vectores.

Una vez que se ha definido un vector, es posible acceder fácilmente a cada una de sus componentes con los comandos de `MATLAB`. Por ejemplo, introdúzcase el siguiente vector.

```
>> x=[10,13,19,23,27,31,39,43,51]
```

Ahora pruébense los siguientes comandos.

```
>> x(2)
>> x(7)
```

Se puede cambiar fácilmente el contenido de una componente.

```
>> x(6)=100
```

Se puede también acceder a un rango de elementos

```
>> x([1,3,5])
>> x(1:3)
>> x(1:2:length(x))
```

3. Operaciones con vectores

Un gran número de operaciones en las que intervienen vectores y escalares se pueden ejecutar con MATLAB.

3.1. Operaciones entre vector y escalar.

Las operaciones entre escalares y vectores son directas. Desde el punto de vista teórico, no se puede sumar un escalar a un vector. Sin embargo, MATLAB sí lo permite. Por ejemplo, si y es un vector, el comando y+2 añadirá 2 a cada componente del vector. Estúdiense las salidas de los siguientes comandos.

```
>> y=1:5
>> y+2
>> y-2
>> 2*y
>> y/2
```

Por supuesto, estas operaciones son igualmente válidas para vectores columna.

```
>> w=(1:3:20)'
>> w+3
>> w-11
>> .1*w
>> w/10
```

3.2. Operaciones entre vectores.

En primer lugar, considérense los siguientes vectores.

```
>> a=1:3
>> b=4:6
```

La adición y sustracción de vectores es natural y fácil. Introdúzcanse los siguientes comandos.[2]

```
>> a,b,a+b
>> a,b,a-b
```

De nuevo, estas operaciones son válidas para vectores columna.

```
>> a=(1:3)',b=(4:6)'
>> a+b,a-b
```

Sin embargo, se pueden obtener resultados no esperados si no se recuerda que MATLAB es un entorno que trabaja con matrices.

```
>> a,b,a*b
```

El último comando devuelve un error porque $*$ es el símbolo de MATLAB para la multiplicación de matrices, y en este caso hay un problema de compatibilidad entre los ordenes de las "matrices" a y b. También pueden ocurrir errores si se intenta añadir vectores de diferente tamaño.

```
>> a=1:3,b=4:7,a+b
```

3.3. Operaciones con componentes.

Para multiplicar los vectores a y b componente a componente, ejecútese el siguiente comando de MATLAB.

[2]Como no aparece punto y coma que suprima la salida, el comando a,b,a+b mostrará primero el vector a, luego el vector b y por último el a+b

```
>> a=(1:3)',b=(4:6)'
>> a,b,a.*b
```

El símbolo $.*$ es el operador de MATLAB para la multiplicación elemento a elemento. La salida se calcula multiplicando las primeras componentes de los vectores `a` y `b`, a continuación las segundas componentes, etc. El operador de MATLAB para la división componente a componente es `./`

```
>> a,b,a./b
```

Para elevar cada componente de un vector a una potencia, úsese `.^`

```
>> a,a.^2
```

3.4. Expresiones más complicadas.

Con un poco de práctica se aprenderá como evaluar expresiones más complejas. Supongamos, por ejemplo, para evaluar la expresión $x^2 - 2x - 3$ para valores de x entre 1 y 10, con incremento de 1 escríbase

```
>> x=1:10
>> y=x.^2-2*x-3
```

Supóngase ahora que se quiere evaluar la expresión $\text{sen}(x)/x$ para valores de x entre -1 y 1 con incrementos de 0,1 unidades.[3]

```
>> x=-1:.1:1
>> y=sin(x)./x
```

Los operadores por componentes también funcionan con vectores columna.

```
>> xdata=(1:10)'
>> xdata.^2
```

[3]Escribiendo `help elfun` se obtiene una lista de las funciones elementales de MATLAB.

Ejercicios de la práctica 1

Ejercicio 1. Escribe el comando `MATLAB` que genera cada uno de los siguientes vectores.

1. $\begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$.
2. $(1, 2, -1, 3)$.
3. Un vector columna que contenga los números impares entre 1 y 1000.
4. Un vector fila que contenga los números pares entre 2 y 1000.

Ejercicio 2. Si `x=0:2:20`, escribe el comando de `MATLAB` que eleva al cuadrado cada componente de `x`.

Ejercicio 3. Si `x=[0,1,4,9,16,25]`, escribe el comando `MATLAB` que calcula la raíz cuadrada de cada componente de `x`.

Ejercicio 4. Si `x=0:.1:1`, escribe el comando de `MATLAB` que eleva cada componente de `x` a 2/3.

Ejercicio 5. Si `x=0:pi/2:2*pi`, escribe el comando `MATLAB` que calcula el coseno de cada componente de `x`.

Ejercicio 6. Si `x=-1:.1:1`, escribe el comando `MATLAB` que calcula el arcoseno de cada componente de `x`.

Ejercicio 7. Si `x=linspace(0,2*pi,1000)`, ¿cuál es la entrada 50 de `x`? ¿Cuál es la longitud de `x`?

Ejercicio 8. Si `k=0:100`, ¿cuál es la entrada número 12 de `y=0.5.^k`?

PRÁCTICA 2

Matrices y MATLAB

En esta práctica se aprenderá a introducir y editar matrices en MATLAB. Se experimentará con algunas funciones de construcción de matrices incorporadas en MATLAB. Se aprenderá a construir matrices a partir de vectores y bloques de matrices.

Prerrequisitos: ninguno.

1. Entrada de matrices

La entrada de matrices en MATLAB es fácil. Escríbase lo siguiente en el indicador de MATLAB.

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

Obsérvese cómo los símbolos de punto y coma indican el final de la fila, mientras que las comas se usan para separar las entradas en la fila. Se pueden usar también espacios para delimitar las entradas de cada fila.

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

1.1. Matrices especiales.

MATLAB tiene una serie de rutinas incorporadas para crear matrices.[1] Es posible crear una matriz de ceros de cualquier tamaño.

```
>> A=zeros(5)
>> B=zeros(3,5)
```

Es fácil crear una matriz de ceros con el mismo tamaño que una dada.

[1]Para obtener una lista de todas las matrices elementales de MATLAB, escríbase `help elmat` en el indicador de MATLAB; para obtener información detallada sobre una en concreto escríbase `help` seguido del tipo de matriz, por ejemplo, `help magic`.

```
>> C=magic(5)
>> D=zeros(size(C))
```

Se pueden crear matrices de unos de manera análoga.

```
>> A=ones(6)
>> B=ones(2,10)
>> C=hilb(5)
>> D=ones(size(C))
```

Cuando se realizan simulaciones en MATLAB es útil construir matrices de números aleatorios. Se puede crear una matriz de números aleatorios con distribución uniforme, cada uno entre 0 y 1, con los siguientes comandos.

```
>> A=rand(6)
>> B=rand(5,3)
```

La multiplicación por escalares es exactamente igual que para vectores.

```
>> C=10*rand(5)
```

MATLAB proporciona unas rutinas para el redondeo de números.

```
>> D=floor(C)
>> D=ceil(C)
>> D=round(C)
>> D=fix(C)
```

La matriz identidad tiene unos en su diagonal principal y ceros en el resto.

```
>> I=eye(5)
```

Se pueden generar otros tipos de matrices diagonales con el comando diag.

```
>> E=diag([1,2,3,4,5])
>> F=diag([1,2,3,4,5],-1)
```

```
>> G=diag(1:5,1)
```

1.2. Trasposición.

El operador de trasposición, que es ’ (comilla simple), tiene el mismo efecto que sobre vectores. Se intercambian filas y columnas.

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

1.3. Elimina la salida.

Recuérdese que finalizando un comando de MATLAB con punto y coma se elimina la salida. Es útil cuando el resultado es grande y se desea ocultarlo.

```
>> K=rand(100);
```

1.4. Espacio de trabajo de MATLAB.

Examínese el espacio de trabajo con el comando whos, o activando la opción “Workspace” del menú “View” de la barra superior.

```
>> whos
```

Obsérvese que aparece el tamaño de cada una de las variables. Por supuesto, se puede obtener el tamaño de la matriz I con

```
>> size(I)
```

2. Indexado de matrices

La siguiente notación es la que se usa para representar una matriz con 3 filas y 3 columnas.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix},$$

o en forma reducida $A = (a_{ij}) \in \mathcal{M}_3(\Bbbk)$, donde $\Bbbk$ es cuerpo (por ejemplo, $\Bbbk = \mathbb{R}$ o $\Bbbk = \mathbb{C}$. El símbolo a_{ij} se refiere a la entrada situada en la fila i y columna j. MATLAB usa una notación similar para representar los elementos de una matriz.

```
%pascal no funciona en Octave

>> A=pascal(5)
>> A(1,2)
>> A(3,4)
```

En general, `A(i,j)` se refiere al elemento de la fila i, columna j de la matriz A. También es fácil cambiar el valor de una entrada.

```
>> A(3,3)=11111
```

2.1. Algo más sobre indexado.

Cuando se indexa una matriz, los subíndices pueden ser vectores. Esta es una herramienta de gran alcance que permite extraer fácilmente una submatriz de una matriz.

```
>> A=magic(6)
>> A([1,2],[3,4,5])
```

La notación `A([1,2],[3,4,5])` referencia a la submatriz formada por los elementos que aparecen en las filas 1 y 2 y en las columnas 3, 4 y 5 de la matriz A.

El comando

```
>> A([1,3,5],[1,2,3,4,5,6])
```

produce una submatriz con las filas 1, 3 y 5 de la matriz A. Si se recuerda que la notación `1:6` representa al vector `[1,2,3,4,5,6]` y que la notación `1:2:6` representa al vector `[1,3,5]`, de este modo se tiene que `A([1:2:6],[1:6])` es equivalente a `A([1,3,5],[1,2,3,4,5,6])`.

```
>> A([1:2:6],[1:6])
```

Si se usa el símbolo dos puntos en lugar de subíndices, se indica todo el rango. Así,

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

produce la primera columna de la matriz A, y

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

genera la tercera fila de la matriz A. En cierto sentido, la notación `A(3,:)` se puede leer como "Tercera fila, todas las columnas." El comando

```
>> A(1:3,:)
```

produce una submatriz compuesta de las tres primeras filas de la matriz A. El comando

```
>> A(:,1:2:6)
```

produce una submatriz compuesta de las columnas 1, 3 y 5 de la matriz A.

3. Construcción de matrices

Con `MATLAB` se pueden crear matrices más complejas a partir de otras matrices y vectores.

3.1. Construcción de matrices con vectores.

Créense tres vectores fila con los comandos

```
>> v1=1:3
>> v2=4:6
>> v3=7:9
```

El comando

```
>> M=[v1;v2;v3]
```

construye una matriz con los vectores `v1, v2` y `v3`, cada uno formando una fila de la matriz M. El comando

```
>> N=[v1,v2,v3]
```

produce un resultado completamente diferente, pero con sentido.

Cámbiense los vectores `v1,v2,v3` en vectores columna con el operador de trasposición.

```
>> v1=v1'
>> v2=v2'
>> v3=v3'
```

El comando

```
>> P=[v1,v2,v3]
```

construye una matriz con los vectores `v1,v2,v3` como columnas de la matriz P. Se puede obtener el mismo resultado con la transpuesta de la matriz M.

```
>> P=M'
```

Téngase en cuenta que **las dimensiones deben coincidir:** cuando se construyen matrices, hay que asegurarse que cada fila y columna tengan el mismo número de elementos. Por ejemplo, la siguiente secuencia de comandos producirá un error.

```
>> w1=1:3;w2=4:6;w3=7:10;
>> Q=[w1;w2;w3]
```

3.2. Construcción de matrices con otras matrices.

Es una cuestión simple aumentar una matriz con un vector fila o columna. Por ejemplo,

```
>> A=[1,2,3,4;5,6,7,8;9,10,11,12]
>> b=[1,1,1]'
>> M=[A,b]
```

es válido, pero

```
>> M=[A;b]
```

no lo es; aunque sí lo es

```
>> c=[1,1,1,1]
>> M=[A;c]
```

Se pueden concatenar dos o más matrices. Así,

```
>> A=magic(3),B=ones(3,4)
>> M=[A,B]
```

es válido, pero

```
>> N=[A;B]
```

no lo es; aunque sí lo es

```
>> C=[1,2,3;4,5,6]
>> P=[A;C]
```

3.3. La imaginación es el límite.

Las capacidades de construir matrices de MATLAB son muy flexibles. Considérese el siguiente ejemplo.

```
>> A=zeros(3),B=ones(3),C=2*ones(3),D=3*ones(3)
>> M=[A,B;C,D]
```

Se puede construir una matriz de Vandermonde de la siguiente manera

```
>> x=[1,2,3,4,5]'
>> N=[ones(size(x)),x,x.^2,x.^3,x.^4]
```

O también matrices por bloques

```
>> B=zeros(8)
>> B(1:3,1:3)=[1,2,3;4,5,6;7,8,9]
>> B(4:8,4:8)=magic(5)
```

Ejercicios de la práctica 2

Ejercicio 1. Escribe los comandos de `MATLAB` que generan las siguientes matrices.

1.) $\begin{pmatrix} 1 & 1 & 3 & 0 \\ -3 & -1 & 6 & 8 \\ 4 & 4 & 10 & -7 \end{pmatrix}$ 2.) $\begin{pmatrix} 10 & 5 & -6 \\ -5 & -8 & -4 \\ -5 & -10 & 3 \\ 8 & 8 & -5 \end{pmatrix}$.

Ejercicio 2. Escribe un solo comando que cree una matriz 3×5 con cada entrada igual a -3.

Ejercicio 3. Crea una matriz de Hilbert con los siguientes comandos.

```
%Format rat no funciona en Octave

>> format rat
>> H=hilb(5)
>> format
```

Escribe una fórmula para la posición (i, j) de la matriz H.

Ejercicio 4. Explica la diferencia entre los comandos `floor, ceil, round` y `fix`. Apoya tu explicación con ejemplos en `MATLAB`.

Ejercicio 5. Escribe un comando `MATLAB` que genere una matriz 4×4 con valores aleatorios enteros entre -5 y 5.

Ejercicio 6. El operador $'$ genera realmente la conjugada traspuesta de una matriz. Para verlo, introduzca la matriz `A=[1,2+i,sqrt(2);3,3-i,sqrt(-1)]`, teclea entonces `A'`. Describe el resultado. ¿Qué ocurre si ponemos `A.'`? Explica la diferencia entre los operadores de trasposición `'` y `.'`.

Ejercicio 7. ¿Cuál es la entrada en fila 5 y columna 7 de la matriz `pascal(10)`?

Ejercicio 8. Sea `T=toeplitz(1:7)`. Escribe un comando de `MATLAB` que

1. coloca las filas pares de la matriz T en una matriz B.
2. coloca las filas impares de la matriz T en una matriz C.
3. coloca la última columna de la matriz T en un vector b.

Ejercicio 9. Sea `x=[0,pi/2,2*pi]`. Construye una matriz, con comandos de `MATLAB`, cuya primera fila es `x`, la segunda fila está formada por el seno cada entrada de `x`, y la tercera fila es el coseno de cada entrada de `x`.

Ejercicio 10. Sea `x=linspace(0,10)`. Construye una matriz, con comandos de `MATLAB`, cuya primera columna sea `x`, la segunda columna esté formada por los cuadrados de cada entrada de `x`, y la tercera columna sea el inverso de cada entrada de `x`.
