Sistemas Inteligentes
Grado en Ingeniería Informática
elisa.guerrero@uca.es
Departamento Ingeniería Informática

PRÁCTICA 1. INTRODUCCIÓN A MATLAB

MATLAB es el nombre abreviado de "MATrix LABoratory". MATLAB es un programa para realizar cálculos numéricos con ***vectores*** y ***matrices***. Como caso particular puede también trabajar con números escalares, tanto reales como complejos. Una de las capacidades más atractivas es la de realizar una amplia variedad de ***gráficos*** en dos y tres dimensiones.

Variables:

1. NO pueden comenzar con un número, aunque si pueden tener números (variable1 es un nombre válido).
2. Las mayúsculas y minúsculas se diferencian en los nombres de variables. (A y a son dos variables diferentes)
3. Los nombres de variables no pueden contener operadores ni puntos. (No es válido usar /, *, -, +, ...)
4. No es necesario definir el tipo de variable o tamaño (si se usa un vector y después se expande, no hay problema)

1. Comenzando a Trabajar

Dado que MATLAB es un lenguaje *interpretado* comenzaremos hablando de expresiones. Supongamos que nos interesa obtener el valor de multiplicar 2 por 3; tenemos dos formas de hacerlo según el uso que queramos darle al resultado de la multiplicación. La primera:

```
>> 2*3
```

(las instrucciones terminan con el retorno de carro)
entonces obtendremos la respuesta

```
ans = 6
```

asignando el resultado de la expresión (6) a una variable por defecto llamada **ans** (por answer), que contiene el resultado de la última operación. La variable ***ans*** puede ser utilizada como operando en la siguiente expresión que se introduzca.

Evalúa:

```
>>  ans*2
```

Si lo que queremos es almacenar el resultado de la multiplicación en una variable:

```
>> a=2*3
```

y entonces MATLAB asignará a la variable **a** el valor 6.

```
>> a
```

Para utilizar cadenas de texto, con la comilla simple:

```
>> Cadena1= 'Inteligencia Artificial'
```

El uso de la comilla doble

```
 >> cadena2="IA"
```

produce:

??? cadena2="IA"
Error: The input character is not valid in MATLAB statements or expressions.

Para MATLAB el ***infinito*** se representa como ***inf*** ó ***Inf***.

```
» 1.0/0.0
Warning: Divide by zero
ans = Inf
```

MATLAB tiene también una representación especial para los resultados que no están definidos como números:

```
» 0/0
Warning: Divide by zero
ans = NaN
» inf/inf
ans =   NaN
```

En ambos casos la respuesta es ***NaN***, que es la abreviatura de ***Not a Number***.

Órdenes del Intérprete

Orden	Descripción
who / whos	Muestra las variables
help	Muestra la ayuda
clc	Limpia la consola
clear	Borra las variables
close	Cierra las ventanas de gráficos
quit	Cierra Matlab

2. VECTORES Y MATRICES

Las matrices y vectores son ***variables*** que tienen ***nombres***. 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 (o se utilizan).
Las matrices se definen por filas; los elementos de una misma fila están separados por ***blancos*** o ***comas***, mientras que las filas están separadas por ***intro*** o por caracteres ***punto y coma*** (;).
Define una matriz **A** de dimensión (3x3):

```
» A=[1 2 3; 4 5 6; 7 8 9]
» B=[]  %% matriz vacía
```

A partir de este momento las matrices **A y B** están disponibles para hacer cualquier tipo de operación con ella. Por ejemplo, una sencilla operación con **A** es hallar su ***matriz traspuesta***. En MATLAB el apóstrofo (') es el símbolo de *trasposición matricial*. Para calcular **A'** (traspuesta de **A**) basta teclear lo siguiente:

```
» A'
ans =
1 4 7
2 5 8
3 6 9
```

3. OPERADOR DOS PUNTOS (:)

Este operador es muy importante en MATLAB y puede usarse de varias formas.
Para especificar **rangos de valores** el formato básico sería:

ValorMínimo:Incremento:ValorMáximo,

Por ejemplo:

```
>> 2:0.5:4
```

denota al vector fila

```
[2 2.5 3 3.5 4]
```

Esta notación de dos puntos es particularmente útil, entre otras cosas, para introducir matrices de grandes dimensiones. Si por ejemplo, se necesita un vector **A** formado por los cien primeros enteros (del 1 al 100), bastará escribir:

****** Si no se especifica el valor del incremento se toma como 1**

```
>> A=1:100;
```

Las siguientes instrucciones generan una "tabla de senos", (**sin** es la función seno).

```
>> x=[0.0:0.1:2*pi];
>> y=sin(x);
```

EJERCICIO 1

Utiliza plot para mostrar gráficamente la función seno(x):
(help plot para estudiar más opciones de esta orden)

plot(x,y,'-r') %%% dibuja la función y=seno(x) trazando una línea roja: '-r'
title('Funcion seno(x)') %%% título de la figura
legend ('y=seno(x)') %% leyenda

4. ACCESO A LOS ELEMENTOS DE UNA MATRIZ

Para acceder a un elemento de la matriz las filas y columnas se indexan de 1 a N filas y de 1 M columnas. Para referenciar una columna o fila completa se utiliza el operador **:** (2 puntos)

EJERCICIO 2

Comprueba que son ciertas las siguientes afirmaciones con la matriz

```
A=[10 22 34; 43 55 64; 76 86 96; 89 90 91]
```

A(4,3) : elemento de la 4ª fila y la 3ª columna

A(1,**:**) : todos los elementos de la 1ª fila

A(**:**,2) : todos los elementos de la 2ª columna

A(2:3, :) : todos los elementos de la 2ª a la 3ª fila y todas las columnas

¿Qué instrucción es necesaria para acceder a los elementos de las filas 1 y 2 de las columnas 2 y 3?

5. OPERADORES BÁSICOS: Para más ayuda sobre las operaciones matriciales en general ver **help ops**.

+ adición o suma – sustracción o resta * multiplicación	.* producto elemento a elemento .\ división elemento a elemento .^ elevar a una potencia elemento a elemento Sqrt : raíz cuadrada	' traspuesta ^ potencia

EJERCICIO 3
Evalúa las siguientes expresiones y describe la función que realizan con las siguientes matrices:
A=[2 3 4; 4 5 6; 5 6 7];
B=[2 2 2; 2 2 2; 2 2 2];

1. ¿Qué tamaño tienen las matrices?, puedes usar las funciones **whos**, o **size(A) size(B)**

2. Evalúa : a
3. Evalúa : A
4. Evalúa : A+B
5. Evalúa : A*B
6. Evalúa : A.*B
7. Evalúa: A'
8. ¿Qué diferencias encuentras entre A.*B y A*B?
9. ¿Qué función realiza el . ?

6. FUNCIONES ESCALARES

Ciertas funciones de MATLAB operan esencialmente sobre escalares y si lo hacen sobre matrices lo hacen elemento a elemento. Veamos algunas de estas funciones (para saber más sobre estas funciones y conocer otras nuevas podemos escribir **help elfun**):

sin (seno) **asin**	**cos** (coseno) **acos**	**tan** (tangente) **atan**	**log2** **log10** **log (neperiano)**
sign(a) -1 si a es negative +1 si a es positivo	**abs** : Valor Absoluto	**round:** redondeo **fix:** truncamiento	**mod(**num, divisor) resto de la división entera entre num y divisor

Por ejemplo:

```
>> x=pi/4;
>> sin(x)
  ans = 0.7071
```

O también:

```
>> x=[0:4; 5:9];
>> sin(x)
  ans =  0          0.8415    0.9093    0.1411    -0.7568
        -0.9589    -0.2794    0.6570    0.9894     0.4121
```

7. FUNCIONES VECTORIALES

Hay funciones de MATLAB que están pensadas para actuar sobre vectores (fila o columna). Veamos algunas de estas funciones (con la instrucción help datafun podemos aprender más sobre estas funciones):

max min	**sum prod cumsum sumprod**	**median mean std** (mediana, media, desviación estándar)	**sort**

EJERCICIO 4

Evalúa las siguientes expresiones y describe la función que realizan con las siguientes matrices:

```
A=[18   2   2; 3   4   6; 1   4   5];
B=[ 2    2   2; 2   2   2; 2   2   2];
```

1. max(A):
2. min(A):
3. sum(B):
4. cumsum(B):
5. prod(B)
6. cumprod(B):
7. median(A):

8. mean(A):
9. std(A):
10. sort(A):

8. FUNCIONES QUE SIRVEN PARA DEFINIR MATRICES O VECTORES

Existen también una serie de funciones disponibles en MATLAB para la construcción de matrices. Algunas de ellas son:

zeros	matriz de ceros
ones	matriz de unos
eye	matriz identidad
rand y randn	*(explicada a continuación)*
magic	matriz mágica

EJERCICIO 5

1. Crea una matriz de ceros de 2 filas por 3 columnas
2. Crea una matriz de unos de 4 filas por 2 columnas
3. Crea una matriz identidad de 4 x 4
4. Evalúa

```
>> m1= magic(3)
>> sum(m1)
```

5. ¿Qué es una matriz mágica? Sea m2 la matriz traspuesta de m1. Realiza la suma por columnas de cada una de ellas por separado. ¿Qué resultado se obtiene?

Mediante la instrucción **rand(n)** (o **rand(m,n)**) MATLAB creará una matriz nxn (ó mxn) con elementos generados aleatoriamente y distribuidos uniformemente entre 0 y 1, por ejemplo

```
>> rand(2,3)
```

Si en lugar de que sus elementos estén distribuidos uniformemente queremos que sigan una distribución normal de media 0 y varianza 1 entonces teclearemos

```
>> randn(2,3)
```

Con la función **rand,** MATLAB recorre una lista muy larga de números (desordenados) de tal manera que cuando se le piden 6 números aleatorios lo que hace es dar 6 números consecutivos de su lista. En ocasiones nos puede interesar que se nos ofrezca repetidas veces una misma secuencia de números aleatorios; para ello le indicaremos que se sitúe en una posición concreta de la lista (por ejemplo, la 3)

```
rand('seed',3)
```

Si cada vez que pidamos una secuencia de números aleatorios ejecutamos antes esta instrucción, nos dará siempre la misma secuencia.

9. ARCHIVOS .M, FUNCIONES Y SCRIPTS

- Cada comando/script en MATLAB es un archivo con extensión .m.
- Los scripts agrupan un conjunto de órdenes de Matlab, y no reciben ni devuelven ningún valor a través de parámetros.

CREACIÓN Y EDICIÓN DE UN ARCHIVO .M.

Para crear un nuevo archivo **.m** abriremos el menú **file** de la barra de herramientas y seleccionaremos **new ▸ M-file**. Se abrirá el editor de MATLAB en el que podremos empezar a escribir las instrucciones que deseemos.

FUNCIONES

1) Las variables declaradas en las funciones son variables **locales**, salvo que especifiquemos que son **globales**.
2) A este tipo de archivos sí se les puede pasar parámetros, haciéndolos más versátiles.
3) El nombre de la función debe coincidir con el nombre del archivo .m

El código básico de una función: nombre_funcion.m en MATLAB sería:

```
function argumentos_salida = nombre_función(argumentos_entrada)
% Comentarios
   Instrucciones que incluyan asignar valores a los argumentos de
salida: argumentos_de_salida = (asignación correspondiente) ;
```

Todo esto ha de ser escrito en un archivo llamado ***nombre_función*.m**. Veamos, por ejemplo, el contenido de un archivo **sino.m:**

```
function s=sino(m)
% s=sino(n) – Devuelve 1 si los elementos de la primera y segunda fila
de m son iguales y 0 si no son iguales.
% ejemplo:
% m = [ 2     4     6     8    10; 2     5     6     8    11]
% s=sino(m)
     s=m(1,:)==m(2,:);  %% == operador de igualdad
end
```

Comprueba la ejecución de esta función desde la línea de órdenes del intérprete, con el ejemplo dado o con otro ejemplo que propongas

*** La función sino está limitada a matrices de al menos dos filas. Modifica la función sino para que muestre un mensaje de error cuando la matriz no tenga dos filas, con la orden: disp('Error, la matriz debe tener dos filas') y que avise cuando tenga más de dos filas.

OPERADORES LÓGICOS

< menor que > mayor que <= menor o igual que >= mayor o igual que **==** igual que **~=** distinto que	**&** and **|** (ALTGR+1) or **~** (ALT+126) not Si en una relación los dos términos comparados son escalares, el resultado será 1 ó 0 dependiendo de si la relación es cierta o falsa, respectivamente.

EXPRESIONES CONDICIONALES Y BUCLES

if (condición) (órdenes 1) **[else,** (órdenes 2)**]** **end**	**if** condición1, (órdenes 1) **elseif** condición2, (órdenes 2) …. **else** (órdenes n) **end**	**switch** expresión, **case** arg1, (órdenes 1) **case** arg2, (órdenes 2) **case** arg3, (órdenes 3) **otherwise** (órdenes n) end

while (condición) (órdenes) **end**	**for** indice=inicio:incremento:final (órdenes) **end**

	Ejemplo: for i=1:2:4 disp('hola') end

******* IMPORTANTE:** Los bucles "for" han de ser evitados siempre que sea posible encontrar una expresión equivalente que resuelva satisfactoriamente el problema, pues requieren más cálculos y, por tanto, más tiempo de cómputo. Por ejemplo, el siguiente código (¿qué produce?):

```
for n=1:10
  x(n)=sin(n*pi/10);
end
```

puede ser reescrito de la siguiente y más efectiva forma:

```
n=1:10;
x=sin(n*pi/10);
```

Y aunque ambos códigos producen idénticos resultados, este último se ejecuta en menos tiempo, es más intuitivo, y de paso, requiere menos tecleado.

MANEJO DE ARCHIVOS

Con la instrucción save es posible almacenar datos (en formato binario) en un archivo con extensión .mat, y con la opción –ascii en formato ascii.

save nombre_archivo var1 var2 …
save ('nombre_archivo', 'var1', 'var2', …)

Para cargar los datos de un archivo .mat se utiliza la instrucción load
load('nombre_archivo')

1. **Guarda todas las variables que actualmente estén en el entorno de Matlab en un archivo que se llame** *Viernes*

2. Ejercicios Propuestos

1. Escribe una función que se llamará ecm que realice el siguiente cálculo entre dos vectores.

$$ecm = \frac{1}{N}\sum_{i=1}^{N}(x_i - z_i)^2$$

Tomará como argumento 2 vectores de dimensión N, y devolverá un único valor, ecm.
Si la función que has obtenido contiene algún bucle, intenta quitarlos utilizando funciones como **sum** y **length.**
*** **length** devuelve la dimensión mayor de una matriz

2. Escribe una función que simule la función **cumsum,** o sea, la suma acumulada de una matriz por filas (** prueba la función cumsum para entender su funcionamiento pero no la utilices en la implementación). Entrada una matriz de NxM, salida una matriz NxM con la suma acumulada por filas.

3. Escribe una función que se llame **estadísticas** y devuelva 2 valores: la media y la desviación estándar de un vector (pero sin utilizar las funciones de Matlab mean y std).

4. ¿Qué hace el siguiente código?

```
>> x=rand(6)
>>  y=find(x>0.8)
>>  z=x(y)
```

**** I = FIND(x) Devuelve los índices de los elementos distintos de 0 en X.
**** I = FIND(A>100), devuelve los índices de A que son mayores que 100

5. Escribe una función que recibe un vector de N elementos y devuelve otro vector con aquellos elementos que son mayores de 0.8 ó menores de 0.3.

6. Escribe una función que se llame **divide** para dividir una matriz de N columnas, en dos subconjuntos. Como argumentos de Entrada:

- **M:** una matriz de 2 filas x N columnas
- **Porcentaje:** valor de 1 a 100 que indica el porcentaje de elementos

Como argumentos de Salida:

- **A :** matriz con las primeras "porcentaje" columnas
- **B:** matriz con las "1-porcentaje" columnas restantes

Además deberá mostrar un gráfico donde la primera fila corresponde al eje x y la segunda fila al eje y. En puntos rojos los elementos de A, en círculos verdes los elementos de B.
Por ejemplo:

```
>>  M=[10 20 30 40 50 60 70 80 90 88
        22 32 42 52 22 43 44 3   3   3]
>> [A,B]=divide(M,30);
```

Asignará a A las 3 primeras columnas y a B las siete restantes:

```
A=[10 20 30
     22 32 42]
B=[40 50 60 70 80 90 88
     52 22 43 44 3  3   3  ]
```