Práctica 1: INTRODUCCIÓN A MATLAB

Cuestiones básicas

Matlab es un programa comercial que se utiliza en muchas universidades, centros de investigación y empresas para llevar a cabo cálculos científicos. El programa **GNU Octave** es muy similar al anterior pero con la ventaja de que es software libre. Ambos están disponibles para distintos sistemas operativos.

El libro "Aprenda Matlab como si estuviera en primero" de la Universidad Politécnica de Madrid lo puedes utilizar como libro de referencia, así como la propia ayuda que trae Matlab en "Help" en la barra de herramientas. También puedes encontrar por Internet mucha ayuda ya que existen muchos usuarios de estos programas en todo el mundo.

Al iniciar Matlab se abren varias ventanas, aunque nosotros vamos a trabajar sólo con dos:

- **la ventana de comandos** donde aparece el prompt del sistema >>
- **la ventana del editor de texto** que usaremos para escribir programas. Esta ventana se puede abrir con el comando edit o con la opción "archivo nuevo" del menú de herramientas de Matlab.

- Matlab está escrito en C.

- El símbolo % se utiliza para los comentarios.

- Matlab/Octave distinguen entre mayúsculas y minúsculas.

Matlab se pueden utilizar como calculadora para llevar a cabo cálculos.

Basta con teclear la orden en la línea de comandos, por ejemplo, si escribimos en la línea de comandos del programa:

```
>>2^3+(1/5)
```

El programa devuelve: 8.2

Sin embargo, si el cálculo se asigna a una variable, el resultado queda guardado en ella:

```
>>x=2+3
```

Para conocer el valor de una variable, basta teclear su nombre:

```
>>x
```

Si se añade un punto y coma (;) al final de la orden, el ordenador no muestra la respuesta:

```
>>y=5*4;
```

pero no por ello deja de realizarse el cálculo.

- Se pueden utilizar las funciones matemáticas habituales.

`abs` –valor absoluto

`sqrt` - raíz cuadrada

`floor` – entero más cercano a la izquierda, por ejemplo: floor(2.7)=2

`ceil` – entero más cercano a la derecha, por ejemplo: ceil(2.7)=3

`exp` - exponencial base e

`log` - logaritmo neperiano

`sin, cos`,…- función seno, coseno, …

Así, por ejemplo, la función coseno,

```
>>cos(pi)
```

-1

o la función exponencial

```
>>exp(1)   % Función exponencial evaluada en 1, es decir, el número e
```

2.7183

- **El usuario puede controlar el número de decimales con que aparece en pantalla el valor de las variables**, sin olvidar que ello no está relacionado con la precisión con la que se hacen los cálculos, sino con el aspecto con que éstos se muestran:

```
>>1/3
ans=
0.3333
>>format long
>>1/3
ans=
0.33333333333333
>>format       % Vuelve al formato estándar
```

Si hacemos:

```
>> 1/3+2/5
```

Matlab nos devuelve 7.333

Si hacemos:

```
>>format rat
>>1/3+2/5
```

Matlab nos devuelve: 11/15

El comando “rat” viene de “rational” que significa número racional.

- **La ayuda de Matlab es bastante útil**: para acceder a la misma basta teclear help. Es recomendable usarlo para obtener una información más precisa sobre la sintaxis y diversas posibilidades de uso de los comandos.

- Si se teclea pwd Octave/Matlab nos indicará cuál es el directorio de trabajo donde está operando Octave/Matlab. Con el comando cd podemos cambiar dicho directorio. Por ejemplo cd: c\

- El comando **whos** nos devuelve las variables actuales en memoria.
Si a, por ejemplo, es una variable, el comando whos a nos indica de qué tipo es y su precisión.

- La orden **clear** borra de memoria todas las variables actualmente en memoria. También se puede utilizar la orden: **clear a** si queremos solamente borrar de memoria la variable *a* dejando el resto.

- Se puede utilizar la orden **clc** para poner en blanco la ventana de comandos del programa.

- La orden **path** nos dice el path, es decir, los sitios donde se van a buscar las funciones o programas.

Vectores y matrices en Matlab/Octave

Los vectores se escriben de la siguiente forma:
$vec = [0\ \ 5\ \ -3]$ o bien $vec = [0,\ 5, -3]$
separando los elementos por espacios en blanco o por comas.
Para referirnos a un elemento concreto de un vector pondremos, por ejemplo, $vec(2)$ y el programa nos devuelve un 5.

La orden $x = 1{:}\,2{:}\,10$ crea un vector con primer elemento igual a 1 y va incrementando de 2 en 2 hasta llegar a 10 como mucho, es decir: $x = [1\ 3\ 5\ 7\ 9]$
La orden x=1:5 crea el vector $x = [1\ 2\ 3\ 4\ 5]$. Es decir, el incremento por defecto es 1.

Las matrices se escriben de la forma: $A = [2\ \ 3\ \ -5;\ 4\ \ 4\ \ 5]$, separando las filas por ;
Para referirnos a un elemento concreto de la matriz anterior pondremos, por ejemplo, A(1,2) y nos devuelve un 3.

Se pueden unir vectores o matrices, por ejemplo, si $A = [2\ 3\ 5]$ y $B = [2\ 5\ 6]$, entonces:

$C = [A;\ B]$ nos devuelve la matriz $C = \begin{pmatrix} 2 & 3 & 5 \\ 2 & 5 & 6 \end{pmatrix}$ mientras que $C = [A\ B]$ nos devuelve el vector $C = (2\ 3\ 5\ 2\ 5\ 6)$.

El comando **length**(a) nos devuelve la longitud de un vector

El comando **size**(A) nos devuelve el tamaño de una matriz.

El comando A=**zeros**(2,4), sirve para crear una matriz de ceros de tamaño 2x4.

El comando A=**ones**(2,4) sirve para crear una matriz de unos de tamaño 2x4.

La orden A^n calcula la potencia de una matriz pero la orden A.^n eleva elemento a elemento.

Por ejemplo:

La orden:

```
A=[1 -1; 2 3];
B=A^2
```

Devuelve: $B = [-1\ \ -4;\ 8\ \ 7]$

Mientras que:

```
A=[1 -1; 2 3];
B=A.^2
```

Devuelve: $B = [\ 1\ \ 1;\ 4\ \ 9]$

Característica especial de Matlab: vectorización

En todos los lenguajes de programación se entienden las expresiones:

```
x=1.5;
y=sin(2*x)
```

Obteniendo:

y =0.1411

Matlab tiene una característica especial y es que también lo hace para vectores, es decir:

```
x=[1.5 3.1 -2.3];
y=sin(2*x)
```

devuelve

y=[0.1411 -0.0831 0.9937]

Entonces, Matlab permite usar expresiones del tipo:

```
x=[1.5 3.1 -2.3];
y=2*x+1
```

Obteniendo

y=[4.0000 7.2000 -3.6000]

Y también expresiones similares.

Sin embargo, no entendería (y nos daría un mensaje de error) lo siguiente:

```
x=[1.5 3.1 -2.3];
y=x^2+1
```

El mensaje de Matlab es: "Matrix must be square".

Es decir, x es un vector fila y Matlab intenta hacer el cuadrado de una matriz que no es cuadrada. Por eso da el mensaje de error. Si queremos hacer la operación elemento a elemento debemos usar la expresión:

```
x=[1.5 3.1 -2.3];
y=x.^2+1
```

Ahora funciona correctamente y nos devuelve:

y=3.2500 10.6100 6.2900

Programación en Matlab/Octave: Scripts o programas

Se puede trabajar en modo consola, como hemos visto hasta ahora, o a través de **scripts o programas**. Un script es un fichero de texto que contiene un conjunto de órdenes. Estos scripts se deben escribir con cualquier editor de texto plano (texto ASCII) y guardado con la extensión .m.

Tanto Matlab como Octave traen sus propios editores de texto.

El fichero debe estar en el directorio de trabajo y para ejecutarlo bastará con escribir en la línea de comandos el nombre del programa.

La orden **path** nos proporciona qué directorios están en el path.

La orden **dir** o también **ls** nos da el contenido del directorio actual.

Por ejemplo, escribe un fichero llamado 'ejemplo.m' en el directorio de trabajo con el siguiente contenido:

```
% Crea un vector
x=0: 2*pi/10: 2*pi;
% Crea otro vector asociado al anterior
y=cos(2*pi*x);
% Muestra el resultado
disp(y);
```

La orden disp( ) lo que hace es mostrar en pantalla el valor de una variable La variable puede ser de cualquier tipo: número, vector, matriz, cadena de caracteres, …

Después escribe en la línea de comandos:

```
>>ejemplo
```

y verás cómo se ejecutan las órdenes que hay dentro del script.

Ejercicio 1

Crea un programa (fichero .m) que te pida dos números usando la orden input de Matlab y nos muestre en pantalla el resultado de la suma de sus cuadrados.

Nota: La orden input se introduce de la forma a=input('introduce un número');

Bucles en Matlab/Octave

Bucle for

Su formato es sencillo de entender, por ejemplo:

```
for i=1: 2: 6
..
end
```

genera un bucle for con el contador i comenzando en 1, incrementándolo de 2 en 2 mientras que i sea menor o igual que 6.

Si sólo se pone:

```
for i=1: 6
..
end
```

se supone que el incremento de contador es 1, es decir, que equivale a: for i=1:1:6

Escribe el script siguiente y mira qué es lo que sale:

```
for i=1:10
fprintf('el valor de %d al cuadrado es %d\n',i,i^2);
end
```

Como ves en el ejemplo anterior, la orden **fprintf** de Matlab se parece mucho a la orden printf de C que estás acostumbrado a usar. Puedes usar %d para enteros, %f para reales, %.4f si quieres que muestre 4 decimales, etc.

Ejemplo

Vamos a "castigar" al ordenador escribiendo 10 veces la frase 'Hola mundo'.

```
for i=1:10
    fprintf('Hola mundo\n');
end
```

También se puede utilizar el código:

```
for i=1:10
    disp('Hola mundo');
end
```

Ejercicio 2

Escribe un script que permita escribir en pantalla todos los números enteros entre 1 y 25 y sus raíces cuadradas.

Ejemplo:

El siguiente programa permite sumar todos los elementos de un vector:

```
v=[2 3 5 -7.1];
suma=0;
for i=1:length(v)
    suma=suma+v(i);
end
fprintf('El resultado es %f\n',suma)
```

Nota: Con la orden de Matlab sum(v) podríamos obtener directamente el resultado, pero lo hemos hecho sin esa función para comprender el funcionamiento del bucle for.

Ejercicio 3

Escribe un programa que permita averiguar el producto de todos los elementos del vector

$[2 \;\; 3 \;\; 5 \;\; -7.1]$

Ejercicio 4

Escribe un programa que permita averiguar la suma de los cuadrados de los números naturales pares comprendidos entre 1 y 100.

Ejemplo:

El siguiente programa permite sumar todos los elementos de una matriz:

```
A=[2 3 5 -7.1; 1 2 3 4];
suma=0;
[nfilas, ncolumnas]=size(A);
for i=1:nfilas
    for j=1:ncolumnas
    suma=suma+A(i,j);
    end
end
fprintf('El resultado es %f\n',suma);
```

Ejercicio 5

Escribe un programa que permita averiguar para la matriz del ejemplo anterior:

- La suma de cada fila
- La suma de cada columna
- La suma de los cuadrados de todos los elementos de la matriz.

Ejercicio 6

Consideremos la sucesión 1, $2x$, $3x^2$,$4x^3$, $5x^4$, …

Siendo $x = 0.5$, averiguar la suma de los 100 primeros términos de esta sucesión.

(Solución: 4)

Ejercicio 7

Modificar el programa anterior para que el usuario entre el valor de x que desea y el número n de sumandos que desea sumar. Por ejemplo, si $x = 0.5$ y $n = 100$ debe devolver, evidentemente, lo mismo que en el ejercicio anterior. Para ello usarás la función de Matlab input.

Ejercicio 8

Supongamos que tenemos una sucesión $\{a_n\}$ de la que sabemos:

$$a_1 = 2\,; \quad a_2 = 3; \qquad a_n = a_{n-1} + \frac{1}{10}a_{n-2} \qquad n \geq 3$$

Queremos averiguar el valor del término que ocupa la posición 100.

Para ello te proporciono dos códigos incompletos que tú debes completar para hacer que funcionen correctamente.

```
%Primera forma de hacerlo usando un vector
a=zeros(1,100);
a(1)=2;
a(2)=3;
for i=3:100
    a(i)=a(i-1)+;
end
fprintf('el término 100 es %f\n',);
```

```
%Segunda forma de hacerlo sin vectores. Sólo con tres variables: a1, a2 y a3
a1=2;a2=3;
for i=3:100
    a3=a2+0.1*a1;
end
fprintf('el término 100 es %f\n',a3);
```

(Solución: $a_{100} = 15790.8$)

Bucle while

El formato es:

while condición

    sentencias;

end

Ejercicio 9

Escribe el siguiente programa y mira el resultado:

```
x=0;
contador=0;
 while (x<=0.95)
    contador=contador+1;
    x=rand(1);
    fprintf('%f\n',x);
 end
fprintf('contador=%d\n',contador);
```

¿Qué es lo que hace?

Observación:

La función x=rand(1) devuelve un número pseudoaleatorio entre 0 y 1 con $0 < x < 1$.

La orden rand(1, 5) devuelve un vector de 5 componentes con números pseudoaleatorios entre 0 y 1.

La orden rand(3, 2) devuelve una matriz de tamaño 3×2 de números pseudoaleatorios entre 0 y 1.

Condicional if en Matlab/Octave

El formato es:

```
if condición lógica
    ...
end
```

También se puede usar:

```
if condición lógica
   ...
else
   ...
end
```

Los operadores relacionales en Matlab/Octave son los siguientes:

= = igual

< menor que

<= menor o igual que

> mayor que

>= mayor o igual que

~ = distinto

Por ejemplo, escribe y ejecuta lo siguiente:

```
x=rand(1);
if x<0.5
fprintf('resultado en la primera mitad\n');
else
fprintf('resultado en la segunda mitad\n'),
end
```

La condición lógica puede estar formada por varias condiciones lógicas simples unidas por distintos conectivos lógicos como:

&& para "y"

|| para "o"

Por ejemplo:

```
if a>=0 && a<=10
    ....
end
```

La condición lógica `a>=0 && a<=10` será cierta siempre que a sea mayor o igual que cero y a sea menor o igual que 10. Por ejemplo, será cierta para $a = 3$, para $a = 9.8$, etc. Pero no será cierta para $a = -1$, para $a = 13$, etc.

Sin embargo, en el caso de que el condicional fuera:

```
if a<=-2 || a>=10
    ....
end
```

la condición lógica será cierta cuando *a* sea menor o igual que -2 o cuando *a* sea mayor o igual que 10: por ejemplo para $a = -7$, para $a = 13$, etc. No será cierta si $a = 5$, si $a = 0$, etc.

La orden break

La orden **break** sirve para parar la ejecución de un script justo en la línea donde se encuentre situada o bien para salirse de un bucle si está insertada dentro de un bucle.

Por ejemplo, supongamos que queremos hacer un programa que nos imprima el cubo de los números naturales entre 1 y 100 siempre que dicho cubo sea menor o igual que 10000.

```
for i=1:100
    resu=i^3;
    if(resu>10000)
        break;
    end
    fprintf('i=%d su cubo es %d\n',i,resu);
end
```

El resultado es que nos imprimirá los cubos de los números entre 1 y 21 ya que el cubo de 22 será mayor que 10000

Ejercicio 10:

Recordar que un número es divisible por otro cuando al hacer la división entera resulta que el resto de la división es cero. Por ejemplo, 8 es divisible por 2, 12 es divisible por 3, etc.

Hay una función en Matlab que nos devuelve el resto de la división entera: mod(a, n) devuelve el resto de la división entera de *a* entre *n*. Por ejemplo, mod(13, 3) devuelve un 1.

Hacer un programa que nos permita averiguar cuántos números naturales hay entre 1 y 1000 que no sean divisibles ni por 2 ni por 3 ni por 5.

Ejercicio 11

Supongamos que deseamos simular el lanzamiento de un dado. Usaremos la orden:

ceil(6*rand(1,100))

a) Explica razonadamente qué hace la orden anterior.
b) Úsala para simular 100 lanzamientos de un dado.
c) Cuenta el número de veces que ha salido un 1, el número de veces que ha salido un 2, etc.

Ejercicio 12

Vamos a simular el lanzamiento de una moneda. Se va a ir lanzando una moneda sucesivas veces hasta que consigamos obtener 20 caras. En ese momento pararemos y deberemos sacar en pantalla el número de tiradas que han sido necesarias.