

DEPARTAMENTO DE CIENCIA E INGENIERÍA DE
MATERIALES E INGENIERÍA QUÍMICA

CURSO ACADÉMICO 2016/2017

Introducción a MATLAB



Universidad
Carlos III de Madrid

INTRODUCCIÓN A MATLAB

1

Índice General

| | |
|---|-----------|
| 1. Introducción | 1 |
| 1.1 Características de MATLAB | 1 |
| 1.2 Acceso a MATLAB desde windows | 2 |
| 1.3 Reglas Generales en MATLAB | 3 |
| 2. Primeros pasos | 4 |
| 2.1 Operaciones Básicas | 4 |
| 2.2 Formato de salida. Comando format | 6 |
| 2.3 El punto y coma ; en la línea de comandos | 6 |
| 2.4 Constantes y Variables | 7 |
| 3. Guardar y Leer Datos en Ficheros | 8 |
| 4. Matrices y vectores | 10 |
| 4.1 Introducción de matrices en la línea de comandos | 10 |
| 4.2 Elementos de matrices y vectores | 11 |
| 4.3 Funciones para la construcción de matrices | 12 |
| 4.4 Notación de dos puntos : | 13 |
| 4.5 Submatrices y Matrices por bloques | 14 |
| 4.5.1 Submatrices | 14 |
| 4.5.2 Matrices por Bloques | 15 |
| 4.6 Operaciones Matriciales y Puzuzados | 16 |
| 4.7 Resolución de Sistemas de Ecuaciones Lineales | 17 |
| 4.8 Otras operaciones sobre matrices | 18 |
| 4.8.1 Operaciones de manipulación de filas y columnas | 19 |
| 5. Gráficos | 22 |
| 5.1 Gráficos 2D | 22 |
| 5.2 Gráficos 2D Múltiples | 23 |
| 5.3 Edición de gráficos | 24 |
| 5.4 Gráficos 3D | 25 |
| 6. Programación | 26 |
| 6.1 Reglas Generales | 26 |
| 6.2 Tipos de m-filer | 26 |
| 6.2.1 Archivos de instrucciones | 27 |
| 6.2.2 Archivos de funciones | 28 |
| 6.3 Estructuras de Control del Flujo del Programa | 30 |
| 6.3.1 Bloque for | 30 |
| 6.3.2 Sentencia if | 31 |
| 6.3.3 Sentencia while | 31 |
| 6.4 Operadores Lógicos y Operadores Relacionales | 32 |
| 7. Bibliografía | 34 |


1 Introducción

MATLAB \Rightarrow MATrix LABoratory

1.1 Características de MATLAB

- Programa de **Cálculo Numérico** y **Cálculo Simbólico**.
 - Se puede considerar como una calculadora programable **muy potente**.
- Programa muy popular entre **estudiantes, ingenieros, técnicos** e **investigadores** debido a sus características:
 - Programa **interactivo**.
 - **Capacidades Gráficas** potentes y sencillas.
 - Posee gran cantidad de **Funciones** de todos los tipos.
 - **Lenguaje de programación** de alto nivel similar a *Fortran, C, Pascal o Basic*, pero **más fácil** de aprender.
- Existen versiones del programa MATLAB desde pequeños **PC's** hasta **superordenadores**.

1.2 Acceso a MATLAB desde windows

1. Pulsar con el botón izquierdo del ratón sobre el botón **Inicio**.
2. Pulsar con el botón izquierdo del ratón sobre la **carpeta** de **Programas**.
3. Pulsar con el botón izquierdo del ratón sobre la **carpeta** de **Matlab** .
4. Pulsar con el botón izquierdo del ratón sobre uno de los **iconos** de **MATLAB**.
5. Aparecerá una ventana con el *prompt* : `>>`

`>>`

6. Crear un directorio

```
>> !mkdir A:\nombre_directorio
>> cd A:\nombre_directorio
```

1.3 Reglas Generales en MATLAB

- MATLAB distingue entre mayúsculas y minúsculas:

```
>> min(2,3)
```

NO es lo mismo que:

```
>> MIN(2,3) % error
```

- Los **espacios en blanco SÍ** tienen significado para matlab \implies separan distintos elementos en una matriz;

```
>> 12
```

```
>> 10e-12
```

no es lo mismo que:

```
>> 1 2 % error
```

```
>> 10 e-12 % error
```

- Los **PARÉNTESIS** `()` y los **CORCHETES** `[]` tienen significados **distintos**.
- Las **Flechas**: \uparrow y \downarrow permiten **recuperar mandatos**.
- Las **Flechas**: \leftarrow y \rightarrow permiten **corregir errores**.
- Para obtener **AYUDA** en el entorno de MATLAB se utiliza el comando **help**:

```
>> help help
```

```
>> help for
```

```
>> help plot
```

2 Primeros pasos

2.1 Operaciones Básicas

| | |
|---|--------------------|
| + | adición |
| - | sustracción |
| * | multiplicación |
| ^ | potenciación |
| \ | división izquierda |
| / | división derecha |

| | | |
|-------|-------|-------|
| exp | log | log10 |
| sin | cos | tan |
| asin | acos | atan |
| abs | sqrt | sign |
| round | floor | ceil |

Ejemplos

```
>> 2 + 3
```

```
ans =
```

```
5
```

```
>> 2/6
```

```
ans =
```

```
0.3333
```

```
>> 3*(log10(14.7) - 4/6)/atan(2.3)
```

```
ans =
```

```
1.2940
```

```
>> 1+2i
```

```
ans =
```

```
1.0000+2.0000i
```

```
>> 2 * 2
```

```
ans =
```

```
4
```

```
>> 2\6
```

```
ans =
```

```
3
```

```
>> abs(4+3j)
```

```
ans =
```

```
5
```

El comando flops

- El comando **flops** cuenta el número de **operaciones en coma flotante** realizadas.
- La instrucción **flops(0)** pone a cero dicho contador.
- **Sumas y restas** cuentan **un flop** en aritmética **real** y **dos flops** en aritmética **compleja**.
- **Multiplicaciones** y divisiones cuentan **un flop** en aritmética **real** y **seis flops** en aritmética **compleja**.
- Las **funciones elementales** cuentan **un flop** para argumentos **reales** y más para complejos (depende de la función).

```
>> flops(0)
```

```
>> sin(pi/6)*exp(3.5)      >> flops
ans =                     ans =
    16.5577                 4
```

```
>> (1+2i)*(1-2i)          >> flops
ans =                     ans =
     5                    10
```

2.2 Formato de salida: Comando format

Podemos cambiar la manera en que los resultados numéricos son presentados usando el comando **format**.

```
>> pi                      >> format long
ans =                      >> pi
    3.1416                  ans =
                               3.14159265358979

>> format short e          >> format long e
>> pi                      >> pi
ans =                      ans =
    3.1416e+00              3.141592643589793e+00

>> format bank             >> format
>> pi                      >> pi
ans =                      ans =
    3.14                    3.1416
```

2.3 El punto y coma ; en la línea de comandos

En la línea de comandos se utiliza el **punto y coma ;** al final de una instrucción para que MATLAB no imprima en pantalla el resultado correspondiente. Esto **NO** quiere decir que la operación no se haya ejecutado.

2.4 Constantes y Variables

Reglas

- Podemos definir *constantes y variables* mediante **nombres**.
- El nombre consiste en una **letra** seguida de otras **letras, dígitos** o **subrayados**, hasta un máximo de **31 caracteres** en total.
- MATLAB distigue entre **MAYÚSCULAS** y **minúsculas**.
- Las variables se pueden borrar con `clear nombre`.

Ejemplos

```
>> a = 2; A = 3;
>> alfa = 30; conf = pi/180;

>> sin(conf*alfa+A*a)
ans =
    0.2381

>> ans^2
ans =
    0.0567
```

Si **no** asignamos un **nombre** a una expresión se crea automáticamente la variable **ans** con la que se puede hacer operaciones posteriores.

3 Grabar y Leer Datos en Ficheros

- La instrucción

```
save fname1 x y z
```

graba las variables **x**, **y** y **z** a un fichero (binario) de nombre **fname1.mat** (*archivos mat o MAT-files*).

- La instrucción

```
load fname2 a b
```

recupera las variables **a** y **b** de un fichero (binario) de nombre **fname2.mat**. En **fname2.mat** podría haber almacenadas más variables aparte de **a** y **b**.

- También es posible grabar y recuperar datos a y desde ficheros ASCII: opción `-ascii` de los comandos `load` y `save`.

Ejemplo

```
>> x = 0:pi/5:2*pi;
>> y = sin(x.^2);
>> t = [ x' y' ];
>> save io.mat t
>> clear t
>> x = t(:,1);
??? Undefined function or variable 't'.
>> load io
>> x = t(:,1);
>> y = t(:,2);
>> plot(x,y)
```

El comando diary

La instrucción

```
diary nombre_fichero
```

se utiliza para crear un diario de la sesión de MATLAB en el archivo (ASCII) **nombre_fichero**. A partir de dicha instrucción y hasta la introducción en la línea de comandos de la instrucción

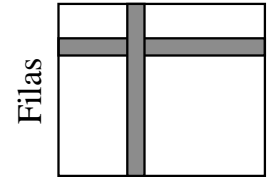
```
diary off
```

todos los comandos que ejecutemos, así como los resultados que devuelva MATLAB (salvo los gráficos) quedarán grabados en el archivo **nombre_fichero**. Luego, podemos abrir y modificar dicho archivo con cualquier editor de texto.

4 Matrices y vectores

4.1 Introducción de matrices en la línea de comandos

Columnas



Matriz: Colección de números ordenados por **filas** y por **columnas**.

- Las *matrices* se definen mediante **corchetes** `[]`.
- Los elementos dentro de una misma fila se separan mediante **comas** o **espacios en blanco**.
- Para indicar el final de una fila y el comienzo de la siguiente se utiliza el **punto y coma**.
- Un **vector fila** de n elementos es una **matriz** $1 \times n$.
- Un **vector columna** de n elementos es una **matriz** $n \times 1$.
- Un **escalar** es una **matriz** 1×1 .

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

```
A =
```

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

```
>> v1=[ 1, 2, 3, 4 ]
```

```
v1=
```

```
1     2     3
```

```
>> v2=[ 1; 2; 3 ]
v2 =
     1
     2
     3

>> size(A)           >> size(v1)
ans =                ans =
     3     3          1     4

>> size(v2)          >> length(v1)
ans =                ans =
     3     1          4
```

4.2 Elementos de matrices y vectores

- Para extraer el **elemento** A_{ij} de una **matriz** A se escribe $A(i,j)$.
- Para extraer el **elemento** v_k de un **vector** v se escribe $v(k)$.

```
>> A(2,3)           >> v1(2)
ans =                ans =
     6                2
```

4.3 Funciones para la construcción de matrices

| | |
|-------------------|--|
| eye(n) | matriz identidad de dimensión $n \times n$ |
| zeros(m,n) | matriz de ceros de dimensión $m \times n$ |
| ones(m,n) | matriz de unos de dimensión $m \times n$ |
| diag(v) | matriz diagonal con diagonal $\{v_k\}_{k=1}^n$ |
| rand(m,n) | matriz aleatoria de dimensión $m \times n$ |

Ejemplos

```
>> a=eye(2)
a =
     1     0
     0     1

>> b=zeros(2,5)
b =
     0     0     0     0     0
     0     0     0     0     0

>> c=rand(2,2)
c =
    0.0579    0.8132
    0.3529    0.0099

>> d=diag([-1, 1])
d =
    -1     0
     0     1
```

4.4 Notación de dos puntos :

El operador `:` es uno de los más importantes en MATLAB. Aparece en diversos contextos:

- Para crear una fila de elementos equidistantes:

```
>> v1= 1:10
v1 =
     1     2     3     4     5     6     7     8     9    10
```

```
>> v2= 100:-7:50
v2 =
    100    93    86    79    65    58    51
```

```
>> v3= 0:pi/4:pi
v3 =
     0    0.7854    1.5708    2.3562    3.1416
```

- Para **extraer** la **fila i-ésima** de una **matriz A** se escribe `A(i,:)`.
- Para **extraer** la **columna j-ésima** de una **matriz A** se escribe `A(:,j)`.
- Para **eliminar** la **fila (columna) i-ésima** de una **matriz A** se escribe `A(i,:)=[]` (`A(:,i)=[]`).
- Para escribir una **matriz A** de dimensiones $m \times n$ como un **vector columna** de mn elementos se escribe `A(:)`

4.5 Submatrices y Matrices por bloques

4.5.1 Submatrices

- La instrucción

$$A([i_1, i_2, \dots, i_r], [j_1, j_2, \dots, j_s])$$

extrae la **submatriz** formada por las **filas** i_1, i_2, \dots, i_r y las **columnas** j_1, j_2, \dots, j_s de la matriz A .

- La instrucción `A([i_1, i_2, ..., i_r], :)` extrae la **submatriz** formada por las **filas** i_1, i_2, \dots, i_r de la matriz A .
- La instrucción `A(:, [j_1, j_2, ..., j_s])` extrae la **submatriz** formada por las **columnas** j_1, j_2, \dots, j_s de la matriz A .

```
>> A = [1:2:5 ; 1:4:9 ; 1:0.1:1.2]
```

```
A =
     1.0000     3.0000     5.0000
     1.0000     5.0000     9.0000
     1.0000     1.1000     1.2000
```

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

```
B =
```

```
     3     5
     5     9
```

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

```
C =
```

```
     1     5     9
     1     3     5
```


4.5.2 Matrices por Bloques

Supongamos que tenemos una matriz A de dimensiones $m \times n$ definida por bloques, por ejemplo,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

donde A_{11} , A_{12} , A_{21} y A_{22} son bloques de dimensiones $p \times r$, $p \times s$, $q \times r$ y $q \times s$ respectivamente, tal que $p + q = m$ y $r + s = n$. Supongamos también que hemos definido estos bloques en MATLAB y los hemos guardado en las variables **A11**, **A12**, **A21** y **A22** respectivamente. Entonces, podemos escribir la matriz A como

```
>> A=[ A11, A12; A21, A22 ];
```

Ejemplos

```
>> A11 = [ 1 2; 3 4]; A12 = eye(2);
```

```
>> A21 = [ -1 0; 0 -1]; A22=ones(2);
```

```
>> A = [ A11 A12; A21 A22 ]
```

```
A =
     1     2     1     0
     3     4     0     1
    -1     0     1     1
     0    -1     1     1
```

4.6 Operaciones Matriciales y Puntuales

| Operadores Matriciales | Operadores Puntuales |
|------------------------|-----------------------|
| + suma | + suma |
| - resta | - resta |
| * multiplicación | .* multiplicación |
| / división derecha | ./ división derecha |
| \ división izquierda | .\ división izquierda |
| ^ potenciación | .^ potenciación |
| ' conjugada traspuesta | .' traspuesta |

- Los **operadores matriciales** son los definidos en el **Álgebra Lineal**.
- Los **operadores puntuales** actúan **elemento a elemento**. Operan entre **matrices** con la **misma dimensión**. El **resultado** es otra **matriz** de igual tamaño.
Si denotamos por $\circ = \{+, -, *, /, \backslash, ^\wedge\}$ a un operador matricial y por $.\circ = \{+, -, .*, ./, .\backslash, .^\wedge\}$ al operador puntual correspondiente. Si A y B son dos matrices cuyos elementos son a_{ij} y b_{ij} respectivamente, entonces los elementos de la matriz $A \circ B$ son $a_{ij} \circ b_{ij}$.
- Las **funciones intrínsecas** de MATLAB (**sin**, **cos**, **tan**, **exp**, **log**, **sqrt**, **abs**, ...) cuando se aplican a una **matriz** actúan **elemento a elemento**.

4.7 Resolución de Sistemas de Ecuaciones Lineales

A Matriz Cuadrada $n \times n$. (**¡e Invertible!**)

$\mathbf{x}=\mathbf{A}\backslash\mathbf{b} \implies$ Solución de $A * x = b$ (**x** y **b** vectores **columna**)

$\mathbf{x}=\mathbf{b}/\mathbf{A} \implies$ Solución de $x * A = b$ (**x** y **b** vectores **fila**)

```
>> A = [ 1, 3, 0; -1, 2, 1; 2, 5, 4 ];  
  
>> b = [ 7; 3 ; 12 ];  
  
>> x = A \ b  
x =  
    1  
    2  
    0
```

4.8 Otras operaciones sobre matrices

| | |
|------------------|--|
| inv(A) | inversa de la matriz cuadrada A |
| pinv(A) | pseudoinversa (Moore-Penrose) de A |
| det(A) | determinante de la matriz cuadrada A |
| rank(A) | rango de la matriz A |
| [n,m] = size(A) | dimensiones de la matriz A |
| tril(A) | parte triangular inferior de A |
| triu(A) | parte triangular superior de A |
| trace(A) | traza de la matriz A |
| null(A) | base ortogonal del núcleo de A |
| orth(A) | base ortogonal de la imagen de A |
| [L,U,P] = lu(A) | factorización LU de A : $P * A = L * U$ |
| R=chol(A) | factorización Cholesky de A : $A = R' * R$ |
| [S,D] = eig(A) | vectores/valores propios de A : $A * S = S * D$ |
| poly(A) | coeficientes del polinomio característico |
| [U,T]=schur(A) | factorización Schur de A : $A = U' * T * U$ |
| [Q,R] = qr(A) | factorización QR de A : $Q * R = A$ |
| [U,S,V] = svd(A) | SVD de A : $U * S * V' = A$ |
| norm(A,p) | norma p=1,2 de A |
| norm(A,inf) | norma ∞ de A |
| norm(A,'fro') | norma de Frobenius de A |
| cond(A,p) | Condicionamiento en norma p de A |

4.8.1 Operaciones de manipulación de filas y columnas

- Para **intercambiar** las **filas i -ésima y j -ésima** de una matriz A se escribe

```
>> A([i j], :) = A([j i], :);
```

- Para **intercambiar** las **columnas i -ésima y j -ésima** de una matriz A se escribe

```
>> A(:, [i j]) = A(:, [j i]);
```

- Para **insertar** una **fila vf** (vector fila) entre las filas k -ésima y $(k+1)$ -ésima de una matriz A $m \times n$ se escribe

```
>> A = [ A(1:k, :); vf; A(k+1:m, :)];
```

- Para **insertar** una **columna vc** (vector columna) entre las columna k -ésima y $(k+1)$ -ésima de una matriz A $m \times n$ se escribe

```
>> A = [ A(:, 1:k), vc, A(:, k+1:n)];
```

Ejemplos con Vectores

```
>> a = [1 2]; b = [3 4];
>> a+b
>> a-b
>> a-1
>> b-2
>> 3*a
>> a/2
>> c = a*b
??? Error using ==> *
Inner matrix dimensions must agree.
>> c = a.*b
c =
     3     8
>> d = a*b'
d =
    11
```

Ejercicios

Define dos vectores filas a y b de 3 elementos.

1. Compara $a.^b$ a^b
2. Compara $a \backslash b$ a/b $a.\backslash b$ $a./b$
3. Compara $a*b'$ $b'*a$ $a*b.'$ $a.'.*b.'$

Ejemplos con Matrices

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

>> C = A*B          >> D = A.*B
C =                  D =
    19    22          5    12
    43    50          21    32

>> E = B.^A          >> F = A.^3
D =                  F =
     5     36          1     8
    343  4096          27    64

>> G = [ 0 pi/6; pi/2 pi];
>> H = sin(G)
H =
    0.0000    0.5000
    1.0000    0.0000
```

Ejercicios

Define una matriz **A** de 2×2 y un vector columna **x** de 2 elementos.

1. Compara $A.^2$ A^2 $A*x$ $x*A$
2. Calcula $\sin(A)$ $\cos(x)*(A+1)$
3. Compara $A \setminus x$ x/A

5 Gráficos

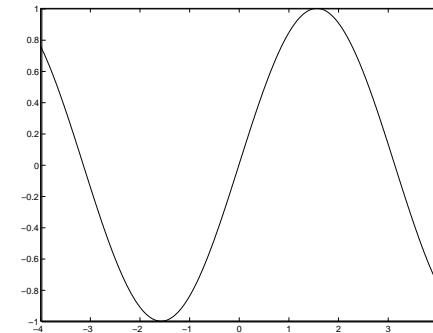
5.1 Gráficos 2D

Representación de los vectores **x** e **y**.

```
>> plot(x,y)
```

Ejemplo

```
>> x = -4:.01:4; y = sin(x); plot(x,y)
```



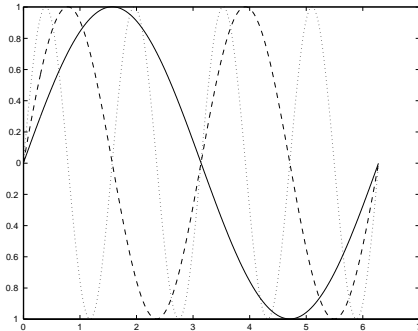
Intentar:

- $y = e^{-x^2}$ sobre el intervalo $[-1.5, 1.5]$
- Gráfica en paramétricas: $x = \cos(3t)$ y $y = \sin(2t)$
 $t = [0, 2\pi]$

5.2 Gráficos 2D Múltiples

Forma 1

```
>> x=0:.01:2*pi;y1=sin(x);
>> y2=sin(2*x);y3=sin(4*x);
>> plot(x,y1,x,y2,'--',x,y3,'.')
```



Forma2

```
>>x=0:.01:2*pi;Y=[sin(x)',sin(2*x)',sin(4*x)'];
>>plot(x,Y)
```

Opciones

- **hold on** permite modificar el último gráfico.
- **hold off** desactiva este modo.

Ejemplo:

```
>> x=0:.01:2*pi;
>> y1=sin(x);y2=sin(2*x);y3=sin(4*x);
>> plot(x,y1)
>> hold on
>> plot(x,y2)
>> plot(x,y3)
>> hold off
```

5.3 Edición de gráficos

- Cambiar el tipo de **línea**, **color**, etc:

```
>> x=0:.01:2*pi; y1=sin(x);
>> y2=sin(2*x); y3=sin(4*x);
>> plot(x,y1,'--',x,y2,':',x,y3,'+')
```

Ver **help plot** para tipos de líneas y colores

- **grid** dibuja una retícula cuadrada.
- Escalado de la ventana gráfica:
`axis([xmin,xmax,ymin,ymax])`
axis, congela el escalado actual.
 Escribiendo **axis** de nuevo volvemos al autoescalado.
- Añadir títulos y comentarios:

| | |
|---------------|--|
| title | título del gráfico |
| xlabel | comentario en el eje x |
| ylabel | comentario en el eje y |
| gtext | texto posicionado interactivamente |
| text | texto posicionado mediante coordenadas |

- Ejemplos:
 1. Escribir: `title('El Titulo')`
 2. Escribir: `gtext('La mancha')` y marcar con el ratón donde se desea que aparezca el comentario.

Ejecutar **help axis**

5.4 Gráficos 3D

- `meshgrid(xx,yy)` crea una retícula a partir de los vectores `xx` e `yy`
- `mesh(xx,yy,z)` representa la función $z(xx,yy)$ sobre la retícula.

Ejemplo

Dibujar $z = e^{-x^2-y^2}$ en el dominio $[-2, 2] \times [-2, 2]$:

Forma 1

```
>> xx = -2:.1:2;
>> yy = xx;
>> [x,y] = meshgrid(xx,yy);
>> z = exp(-x.^2 - y.^2);
>> mesh(z)
```

Forma 2

```
>> [x,y] = meshgrid(-2:.1:2, -2:.1:2);
>> z = exp(-x.^2 - y.^2);
>> mesh(z)
```

Consultar la ayuda para otras funciones gráficas como: `plot3`, `mesh`, y `surf`.

6 Programación

MATLAB permite ejecutar ficheros de instrucciones llamados **m-files**.

`nombre.m`

6.1 Reglas Generales

- Los ficheros deben ser formato *ASCII*.
NUNCA formato Word, WordPerfect, Write, etc.
- MATLAB debe conocer el directorio de trabajo (donde están los ficheros). Para ello utilizar el comando:

```
>> cd nombre_directorio
```

- El símbolo porcentaje `%` se utiliza para **comentarios**. Es importante comentar los ficheros indicando su misión.

6.2 Tipos de m-files

- Archivos de Instrucciones
- Archivos de Funciones

6.2.1 Archivos de instrucciones

- Secuencia de instrucciones dentro de un *m-file*.
- Para ejecutar uno llamado **nombre.m** : `>> nombre`.
- Son muy **útiles**, ya que permiten depurar y reutilizar sentencias **fácilmente**.
- Las líneas de comentario **%** serán mostradas al escribir:

```
>> help nombre_fichero
```

Ejemplo : Fichero figura.m

```
% Fichero ejemplo de m-files
% Tipo Instrucciones
% Genera la figura de 5.2

x=0:.01:2*pi;y1=sin(x);

y2=sin(2*x);y3=sin(4*x);

plot(x,y1,x,y2,'--',x,y3,'. ')

xlabel('X'); ylabel('Y')
title('Grafico multiple')
```

```
>> figura
```

6.2.2 Archivos de funciones

- Estos ficheros crean **nuevas funciones** definidas por el usuario.
- Una vez creadas, pueden utilizarse como una función intrínseca de MATLAB.

Reglas y Consejos

- Fichero ASCII con extensión **.m** (**m-file**).
- Empezar **siempre** con **comentarios (%)**.
- La primera palabra después de los comentarios debe ser: **function**
- El nombre de la función debe ser el **mismo** que el nombre del fichero **sin** extensión **.m**
- Los **parámetros de entrada** deben ser los argumentos de la función, encerrados entre **paréntesis ()**.
- Los **parámetros de salida** van **delante** del nombre de la función.
- Se pueden poner **líneas en blanco** en cualquier sitio.
- Las variables definidas **dentro** del fichero son **locales**, es decir, sólo valen dentro de la función, fuera **no** existen.
- Los nombres de los parámetros de entrada y salida son **variables mudas**, es decir, su nombre puede ser **cualquiera**.

Ejemplo 1

```
% Fichero: rand10.m
% rand10(m,n) produce una matriz m x n
% de numeros aleatorios enteros entre 0 y 9

function a = rand10(m,n)
a = floor(10*rand(m,n));
```

Llamada

```
>> matriz=rand10(3,4)
```

Ejemplo 2

```
% Fichero: sr2.m
% Funcion ejemplo de varios parametros
% de entrada y salida.
% Las dos entradas x e y son dos numeros.
% Las dos salidas s y r son la suma y la resta
% de sus cuadrados respectivamente
% [s,r]=sr2(x,y)

function [out1,out2] = sr2(in1,in2)
out1=in1.^2+in2.^2;
out2=in1.^1-in2.^2;
```

Llamada

```
>> [s,r]=sr2(3,4)
```

6.3 Estructuras de Control del Flujo del Programa

Permiten cambiar el orden de ejecución **secuencial** de las sentencias (una detrás de otra) en un programa.

6.3.1 Bucles for

Repite un conjunto de instrucciones un determinado número de veces.

```
for i= vector fila de índices
    instrucciones(i)
end
```

```
x=[]; for i = 1:n, x(i)=i^2, end
ó
```

```
x = [];
for i = 1:n
    x(i) = i^2;
end
```

Bucles Anidados

```
for i = 1:m
    for j = 1:n
        H(i, j) = 1/(i+j-1);
    end
end
```


6.3.2 Sentencia if

Las instrucciones se ejecutarán sólo si la relación es **cierta**

```
if relación
    instrucciones
end
```

Expresiones más Complicadas

```
if n < 0
    paridad = 0;
elseif rem(n,2) == 0
    paridad = 2;
else
    paridad = 1;
end
```

6.3.3 Sentencia while

Ejecuta unas instrucciones **mientras** la relación sea **cierta**.

```
while relación
    instrucciones
end
```

6.4 Operadores Lógicos y Operadores Relacionales

• Operadores Relacionales

| | |
|----------------------|----------------------|
| < menor que | > mayor que |
| <= menor o igual que | >= mayor o igual que |
| == igual | ~= no igual. |

• Operadores Lógicos

| | | |
|-----|---|------|
| & y | o | ~ no |
|-----|---|------|

- Se utilizan con **for**, **if** y **while**.
- Se pueden utilizar con escalares o matrices.
Cuando se utilizan matrices, el operador actúa componente a componente.

```
>> A =rand10(3,5)
```

A =

```
1    6    7    2    0
8    8    9    4    4
5    3    1    6    1
```

```
>> B= A < 5*ones(3,5)
```

B =

```
1    0    0    1    1
0    0    0    1    1
0    1    1    0    1
```

```
>> p= sum(B(:))/length(A(:))
```

p =

```
0.5333
```

Programa Ejemplo

```
function [P,nops] = mult(A,B)

% [P,nops] = mult(A,B) calcula el producto de
% matrices P = A*B, con A y B de dimensiones
% adecuadas (si no, se obtiene un mensaje de
% error) y el numero de operaciones en coma
% flotante 'nops' realizadas

[mA,nA] = size(A);
[mB,nB] = size(B);

if nA~=mB
    disp('Matrices de dimensiones inadecuadas')
else
    P = zeros(mA,nB);
    flops(0);
    for i=1:mA
        for j=1:nB
            for k=1:nA
                P(i,j) = P(i,j) + A(i,k)*B(k,j);
            end
        end
    end
    nops = flops;
```

7 Bibliografía

- D. Hanselman and B. Littefield, *The Student edition of MATLAB: version 4*, Prentice-Hall, 1995.
- A. Biran and M. Breiner, *Matlab for Engineers*, Addison-Wesley, 1995.
- Eva Pärt-Enander, Anders Sjöberg, Bo Melin and Pernilla Isaksson, *The Matlab Handbook*, Addison-Wesley, 1996.