

Tema 1: Introducción a Matlab

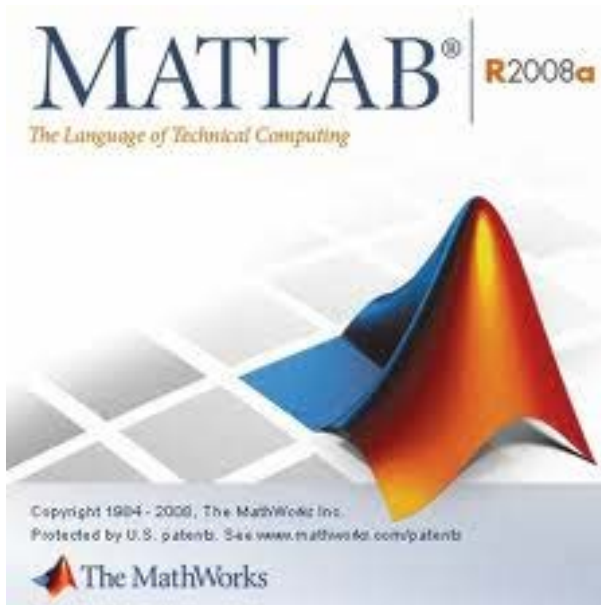
Reconocimiento de patrones - Grado en Ingeniería Informática

Universidad de Cádiz

Curso 2017/ 2018



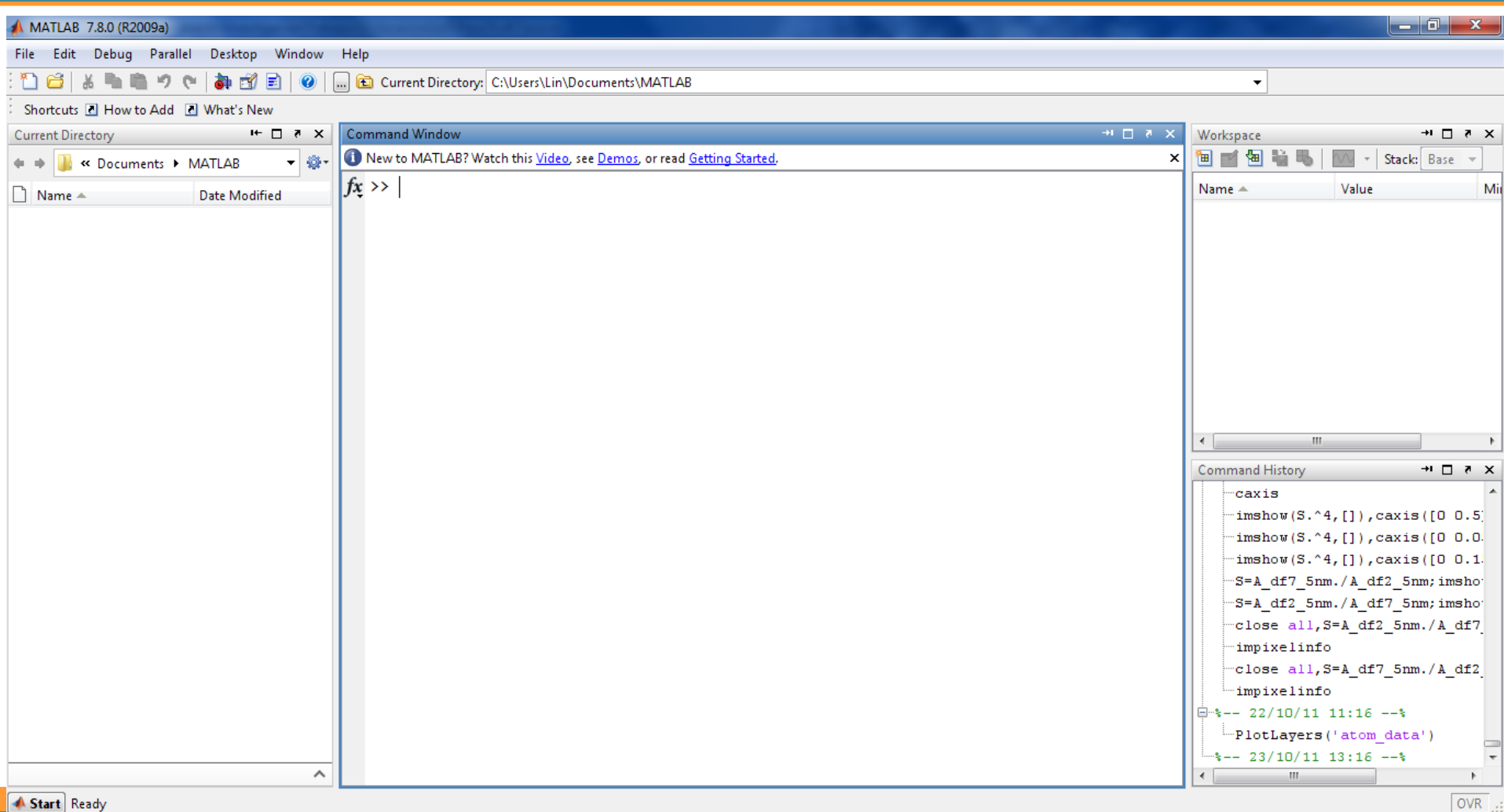
¿Qué es?



☀ Matlab = Matrix Laboratory

☀ **Definición oficial (©The Mathworks Inc.):** "MATLAB: A technical computing environment for high-performance numeric computation and visualization"

Matlab



¿Por qué usaremos Matlab?

- ☀ Es muy fácil de aprender
- ☀ Es muy potente
- ☀ Es fácilmente ampliable
- ☀ Permite trabajar con números complejos y matrices directamente
- ☀ Ofrece acceso a las siguientes herramientas:
 - ☀ Cálculo numérico y matricial
 - ☀ Gráficos en 2D y 3D
 - ☀ Funciones específicas de un área determinada

Matlab como calculadora

```
» 3.5*6.2
```

```
ans =
```

```
21.7000
```

```
» 13+(2-5)*sqrt(16)
```

```
ans =
```

```
1.0000
```

```
»
```

Variables y expresiones

- ☀ Una **variable** es un objeto (realmente una **dirección de memoria**) con un **nombre** que permite **almacenar un valor**.
- ☀ Se distingue entre mayúsculas y minúsculas.
- ☀ El primer carácter de una variable debe ser una letra (A-Z, a-z)
- ☀ El resto pueden ser letras, dígitos y / o el carácter _
- ☀ Hasta 31 caracteres por cada variable

Variables y expresiones

⚙ Tipos de variables

Numéricas

- Reales
- Complejos
- Valores especiales
 - Infinito
 - NaN

Caracteres

- Un único carácter
- Cadenas de caracteres

Datos lógicos

- Verdadero
- Falso

Variables y expresiones

- ⚙ La **variables numéricas** permiten almacenar número reales y complejos.

variable = valor

- ⚙ Es posible operar con valores **Inf** (Infinito) y **NaN** (Not a Number = indeterminado)

Variables y expresiones

- ☀ Las **variables** de tipo **carácter** almacenan un solo carácter. Se utilizan poco, ya que casi no se puede realizar ninguna operación con ellas.
- ☀ Una **cadena de caracteres** permite almacenar una secuencia de caracteres. Se usan con mucha frecuencia.
- ☀ Usaremos comillas simples para definir una cadena.

Variables y expresiones

- ☀ Una **variable lógica** puede tomar dos valores, CIERTO (ó **TRUE**) y FALSO (ó **FALSE**)
- ☀ IMPORTANTE :
 - ☀ Un valor iguala 0 se considera FALSO
 - ☀ Cualquier otro valor se considera CIERTO

Operadores aritméticos, lógicos y relacionales

⚙ OPERADORES ARITMÉTICOS:

Operador	Descripción
+	Suma
-	Resta
*	Producto
/	División
^	Potencia
=	Asignación

Operadores aritméticos, lógicos y relacionales

⚙ OPERADORES LÓGICOS:

Operador	Descripción
&	And
	Or
~	Not

A	B	A & B
V	V	V
V	F	F
F	V	F
F	F	F

A	B	A B
V	V	V
V	F	V
F	V	V
F	F	F

A	~A
V	F
F	V

Operadores aritméticos, lógicos y relacionales

⚙ OPERADORES RELACIONALES:

Operador	Descripción
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual que
~=	Distinto de

Variables y expresiones

⚙ EJEMPLOS

⚙ Número Reales:

```
>> a = sqrt(3)
```

```
a =
```

```
1.7321
```

```
>> b = a - 1;
```

```
>> b
```

```
b =
```

```
0.7321
```

```
>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x1	8	double array

Variables y expresiones

⚙ EJEMPLOS

⚙ Número Complejos:

```
>> c = 3-2*i
```

```
c =
```

```
3.0000 - 2.0000i
```

```
>> c^2
```

```
ans =
```

```
5.0000 - 12.0000i
```

Variables y expresiones

⚙ EJEMPLOS

⚙ Infinito e indeterminaciones:

```
>> a = 3/0
```

```
a =
```

```
Inf
```

```
>> b = a - a
```

```
b =
```

```
NaN
```


Variables y expresiones

⚙ EJEMPLOS

⚙ Cadenas de caracteres:

```
>> a = 'Juan';
```

```
>> b = ' Lopez';
```

```
>> c = [a b];
```

```
>> c
```

```
c =
```

```
Juan Lopez
```

Variables y expresiones

- ☀ Usaremos la función `disp` para mostrar el texto contenido en una variable
- ☀ Usaremos la función `num2str` y `str2num` para convertir números a cadenas y viceversa
- ☀ Usaremos la función `error` para mostrar un texto de error y terminar la operación

Variables y expresiones

⚙ EJEMPLOS

⚙ Datos lógicos:

```
>> a = (3>2)
```

```
a =
```

```
1
```

```
>> b = (a<=0)
```

```
b =
```

```
0
```

Variables predefinidas

☀ Las más importantes son :

pi 3.1415

i , j sqrt(-1)

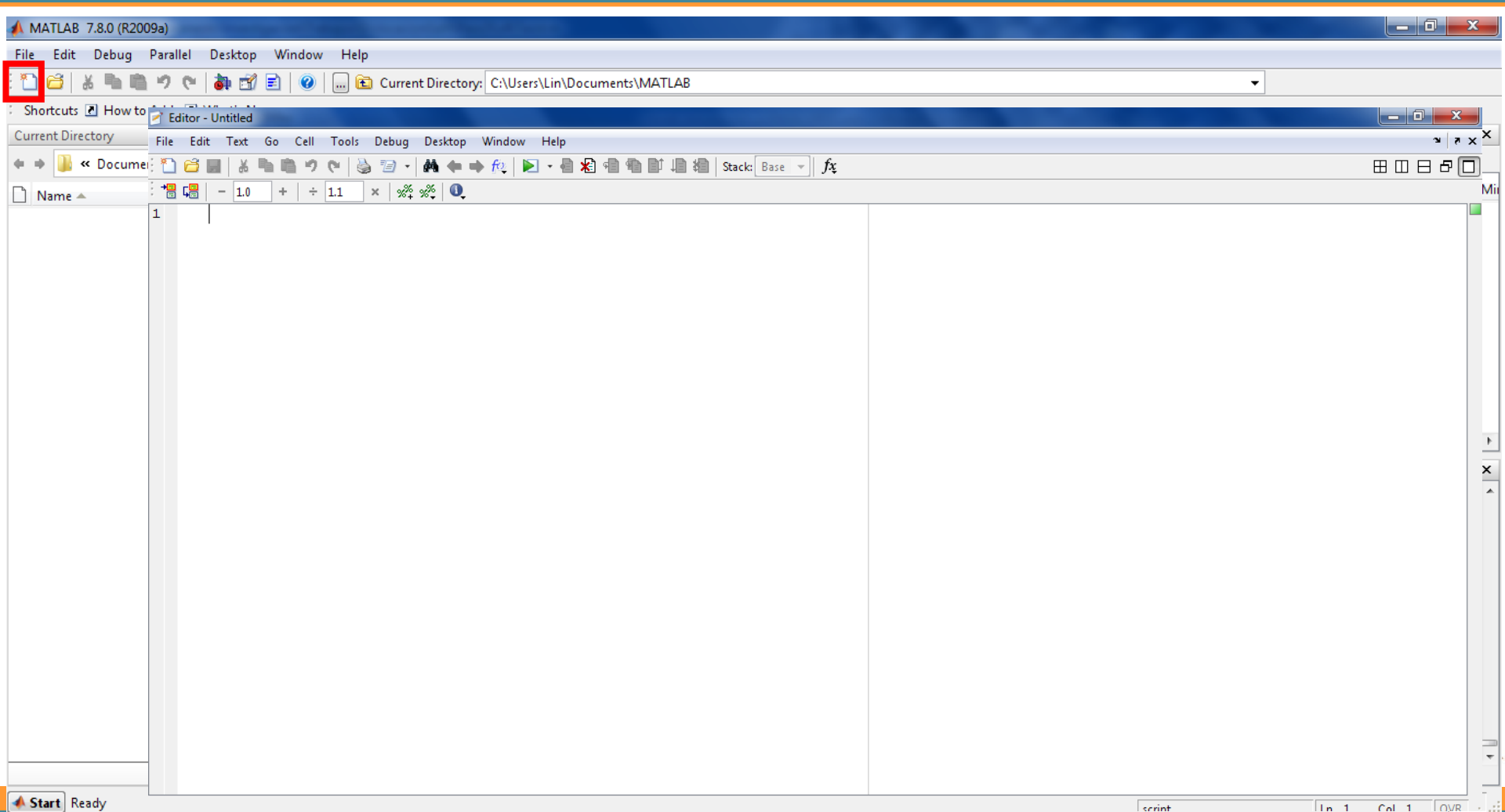
Inf infinito

NaN not-a-number

eps número positivo extremadamente pequeño (epsilon)

☀ Si se asigna un valor a una variable predefinida, ésta pierde su valor anterior

Programando en un script



Programando en un script

⚙️ MI PRIMER PROGRAMA: “Hola mundo”

The screenshot displays the MATLAB 7.8.0 (R2009a) environment. The Editor window shows a script with the command `disp('Hola mundo')`. The Command Window shows the output `Hola mundo`. The File Explorer window shows the current directory `C:\Users\Lin\Desktop` with a list of files and folders.

Editor - Untitled*

File Edit Text Go Cell Tools Debug Desktop Window Help

1 `disp('Hola mundo')`

Save

Guardar en: MATLAB

Nombre:

Tipo:

MATLAB 7.8.0 (R2009a)

File Edit Debug Parallel Desktop Window Help

Current Directory: C:\Users\Lin\Desktop

Shortcuts How to Add What's New

Current Directory

Name	Date Modified
ArticulosMailPete	6/10/11 12:47
David	3/08/11 11:31
GroupMeeting	7/06/11 17:54
Libros_HansKung	16/08/11 11:16
Para MariPaceilla (que ...	22/06/11 15:48
1199646622_67201112536...	9/09/11 12:35
David.rar	3/08/11 17:49
desktop.ini	21/06/11 9:33
Ej_Hola.m	23/10/11 13:32
Entrevista en PuntoRadio...	9/06/11 18:33
facturacionVueltaFinal_2...	18/07/11 20:03
Presentacion York buena...	7/04/11 13:13

Command Window

New to MATLAB? Watch this [Video](#), see [Den](#)

Hola mundo
fx >> |

Funciones trigonométricas

Función	Descripción
<code>sin()</code>	Seno de un ángulo en radianes
<code>cos()</code>	Coseno de un ángulo en radianes
<code>tan()</code>	Tangente de un ángulo en radianes
<code>sinh()</code>	Seno hiperbólico
<code>cosh()</code>	Coseno hiperbólico
<code>tanh()</code>	Tangente hiperbólica
<code>asin()</code>	Inversa del seno
<code>acos()</code>	Inversa del coseno
<code>atan()</code>	Inversa de la tangente
<code>asinh()</code>	Inversa del seno hiperbólico
<code>acosh()</code>	Inversa del coseno hiperbólico
<code>atanh()</code>	Inversa de la tangente hiperbólica

Funciones trigonométricas

⚙ EJEMPLOS:

```
>> sin(pi)
```

```
ans =
```

```
1.2246e-016
```

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

```
ans =
```

```
-1
```

```
>> tan(pi)
```

```
ans =
```

```
1.2246e-016
```


Funciones exponenciales

Función	Descripción
<code>exp()</code>	Calcula e^x
<code>log()</code>	Logaritmo neperiano
<code>log10()</code>	Logaritmo en base 10

⚙ EJEMPLOS:

```
>> exp(2)
```

```
ans =
```

```
7.3891
```

```
>> log(2)
```

```
ans =
```

```
0.6931
```

```
>> log10(2)
```

```
ans =
```

```
0.3010
```

Funciones aritméticas

Función	Descripción
<code>sqrt()</code>	Raíz cuadrada
<code>rem()</code>	Resto de una división
<code>abs()</code>	Valor absoluto

⚙ EJEMPLOS:

```
>> sqrt(4)
```

```
ans =  
  
2
```

```
>> rem(5,3)
```

```
ans =  
  
2
```

```
>> abs(-5)
```

```
ans =  
  
5
```

Funciones de redondeo

Función	Descripción
<code>floor()</code>	Redondeo al entero más bajo
<code>ceil()</code>	Redondeo al entero más alto
<code>fix()</code>	Redondeo al entero más bajo

⚙ EJEMPLOS:

```
>> floor(2.9)
```

```
ans =  
  
2
```

```
>> ceil(2.9)
```

```
ans =  
  
3
```

```
>> fix(2.9)
```

```
ans =  
  
2
```

Funciones sobre complejos

Función	Descripción
<code>real()</code>	Parte real de un n° complejo
<code>imag()</code>	Parte imaginaria de un n° complejo
<code>conj()</code>	Conjugado de un n° complejo
<code>angle()</code>	Ángulo del n° complejo

☀ EJEMPLOS:

```
>> real(2+3*i)
```

```
ans =
```

```
2
```

```
>> imag(2+3*i)
```

```
ans =
```

```
3
```

```
>> conj(2+3*i)
```

```
ans =
```

```
2 - 3.0000i
```

```
>> angle(2+3*i)
```

```
ans =
```

```
0.9828
```

Comandos sobre consola

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

Instrucciones condicionales

☀ Instrucciones IF - ELSE

☀ Instrucciones SWITCH -CASE

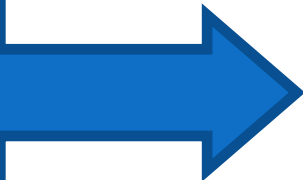
Instrucciones IF - ELSE

```
if condición,  
.....  
end
```



```
if a > 3,  
    b = a + 2;  
end
```

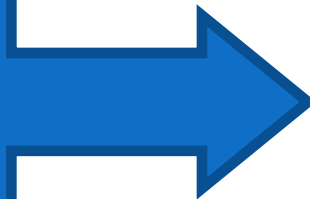
```
if condición,  
.....  
else  
.....  
end
```



```
if a > 3,  
    b = a + 2;  
else  
    b = a + 1;  
end
```

Instrucciones IF - ELSE

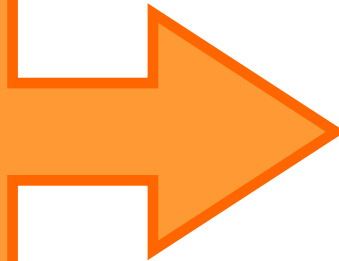
```
if condición1,  
    .....  
elseif condición2,  
    .....  
else  
    .....  
end
```



```
if a > 3,  
    b = a + 2;  
elseif a < 5,  
    b = a + 1;  
else  
    b = a - 2;  
end
```


Instrucciones SWITCH -CASE

```
switch expresión,  
  case arg1,  
    .....  
  case arg2,  
    .....  
  case arg3,  
    .....  
  otherwise  
    .....  
end
```



```
switch a,  
  case 1,  
    disp('A vale 1')  
  case 2,  
    disp('A vale 2')  
  case 3,  
    disp('A vale 3')  
  otherwise  
    disp('Otro')  
end
```

Instrucciones repetitivas

☀ Bucle *while*

☀ Bucle *for*

Bucle *while*

```
while condición,  
    .....  
end
```

Es una estructura cuyo propósito es repetir un bloque de código mientras una condición se mantenga verdadera.

```
i = 1;  
while i < 10,  
    i = i + 1;  
end
```

Bucle *for*

```
for var = Inicio:STEP:Fin,  
    .....  
end
```

Es una estructura de control en la que se puede indicar el número mínimo de iteraciones.

☀ Si STEP = 1, se puede omitir.

☀ **Diferencia entre *for* y *while*:** cuando puede determinarse cuántas vueltas da el bucle usamos *for*, en otro caso, usamos *while*.

```
x = 1;  
for i = 1:6,  
    x = x*i;  
end
```

Funciones: ¿Qué son? ¿Para qué se usan?

- ☀ Las funciones son un grupo de instrucciones bajo el mismo nombre que resuelven un problema concreto.
- ☀ Sirven para solucionar un problema mediante la aplicación del paradigma *Divide y vencerás*.
- ☀ Variables locales
- ☀ Argumentos. Paso por valor
- ☀ No hace falta 'return'
- ☀ Terminan con 'end'

Estructura de una función

```
function [s1, s2, ... , sn =] nombre_fun (e1, e2, ..., en)

%
% Comentarios de ayuda de la función
%

Instrucción 1
Instrucción 2
Instrucción 3
...
Instrucción n
```

Estructura de una función

⚙ EJEMPLO:

```
function r = Cuadrado (n)
    %Función que calcula el cuadrado de un número
    %que se le pasa como parámetro. Ejemplo: r =
    %Cuadrado(3) devolvería en r el valor 9.

    r = n ^ 2;
```

⚙ USO:

```
>> r = Cuadrado(3)

r =

    9
```

Nombre del *script* que contiene a la función

- ⚙ **IMPORTANTE:** El nombre del *script* de Matlab que contiene a la función que implementemos debe ser el mismo que el de dicha función.
- ⚙ **EJEMPLO:** Si la función se llama *Cuadrado* el script debe llamarse *Cuadrado.m*

Vector: ¿Qué son? ¿Para qué se usan?

- ☀ Un **vector** es una zona de almacenamiento continuo que **contiene** una serie de **elementos** del **mismo tipo** bajo el **mismo nombre**.

Índices del vector	1	2	3	4	5	6	7	8	9	10	11	
Elementos del vector	2	5	7	18	9	1	4	5	9	3	12	V

- ☀ Se utilizan en situaciones donde el acceso a los datos se hace de forma aleatoria u ordenada indistintamente.

Definición explícita de un vector

- ⚙ Un vector se define utilizando corchetes.
- ⚙ Podemos tener **vectores filas** y **vectores columna**.
 - ⚙ Los elementos de un **vector fila** se separan por un **espacio** o por una **coma**.
 - ⚙ Los elementos de un **vector columna** se separan por **punto y coma**.

⚙ EJEMPLO:

```
>> F = [1 3 5 7]
```

```
F =
```

```
    1    3    5    7
```

```
>> C = [2;4;6]
```

```
C =
```

```
    2
```

```
    4
```

```
    6
```

Definición implícita de un vector

☀ Se utiliza la siguiente notación:

inicio : intervalo : fin

☀ Si solo se especifican dos parámetros, se supone que son **inicio : fin** y que el intervalo es 1.

☀ EJEMPLOS:

```
>> V = 1:4
```

V =

1 2 3 4

```
>> M = 5 : -2 : 1
```

M =

5 3 1

Acceso a los elementos de un vector

- ☀ Para acceder a los elementos de un vector se utilizan los paréntesis.

- ☀ **EJEMPLO:**

```
>> A = 2:2:10
```

```
A =
```

```
     2     4     6     8    10
```

```
>> A(3)
```

```
ans =
```

```
     6
```

```
>> A(1:2:end)
```

```
ans =
```

```
     2     6    10
```

El operador ':'

- ☀ Permite extraer todos los elementos de cualquier vector en un vector columna.

- ☀ Significa "todos los elementos".

- ☀ **EJEMPLO:**

```
>> V = 1:4;
```

```
>> V(:)
```

```
ans =
```

```
1  
2  
3  
4
```

Operaciones típicas sobre vectores

Función	Significado
'	Transpuesta del vector
sum()	Suma los elementos del vector
prod()	Calcula el producto de los elementos de un vector
mean()	Media de los elementos de un vector
length()	Longitud de un vector
plot()	Dibuja los elementos de un vector
max()	Calcula el elemento máximo de un vector
min()	Calcula el elemento mínimo de un vector
sort()	Ordena los elementos de un vector
cumsum()	Suma acumulada de los elementos de un vector
cumprod()	Producto acumulado de los elementos de un vector
fliplr()	Intercambia los elementos de izquierda a derecha
flipud()	Intercambia los elementos de arriba a abajo

Operaciones típicas sobre vectores

⚙ EJEMPLOS:

```
>> A = 1:5
```

```
A =
```

```
    1    2    3    4    5
```

```
>> A'
```

```
ans =
```

```
    1
```

```
    2
```

```
    3
```

```
    4
```

```
    5
```

```
>> sum(A)
```

```
ans =
```

```
    15
```

```
>> prod(A)
```

```
ans =
```

```
   120
```

```
>> mean(A)
```

```
ans =
```

```
    3
```

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

```
ans =
```

```
    5
```

Operaciones típicas sobre vectores

⚙ EJEMPLOS:

```
>> max(A)
ans =
     5
>> min(A)
ans =
     1
>> B = [2 4 1 6 3]
B =
     2     4     1     6     3
>> sort(B)
ans =
     1     2     3     4     6
```

```
>> cumsum(A)
ans =
     1     3     6    10    15
>> cumprod(A)
ans =
     1     2     6    24   120
>> fliplr(A)
ans =
     5     4     3     2     1
>> flipud(A(1:2)')
ans =
     2
     1
```


El valor *empty*

⚙ Cuando la **dimensión** de un vector es **cero** se dice que se encuentra en estado *empty*.

⚙ **EJEMPLO:**

```
>> a = [];
```

```
>> isempty(a)
```

```
ans =
```

```
1
```

La instrucción *find*

- ☀ Busca las posiciones de los valores que cumplen una condición en un vector.
- ☀ Si no se especifica la condición, *find* buscará los valores distintos de cero en el vector.

☀ EJEMPLOS:

```
>> a = [1 0 5 9 5 2 0 3];
```

```
>> find(a)
```

```
ans =
```

1

3

4

5

6

8

La instrucción *find*

⚙ EJEMPLOS:

```
>> find(a>3)
```

```
ans =
```

```
3    4    5
```

```
>> find(a==2)
```

```
ans =
```

```
6
```

Matrices: ¿Qué son? ¿Para qué se usan?

- ⚙ Una **matriz** es una **extensión** del concepto de **vector**.
- ⚙ Desde el punto de vista lógico una **matriz** se puede ver como un **conjunto** de **elementos** ordenados en **fila**.
- ⚙ Se puede considerar que todas las matrices son de una dimensión, la dimensión principal, pero los **elementos** de dicha **fila** pueden ser a su vez **matrices**.
- ⚙ Las matrices son **multidimensionales**, aunque las más fáciles de visualizar son los de una (vector), dos y tres dimensiones.

¿Qué son? ¿Para qué se usan?

	1	2	3	4	5	
1	5	6	2	9	11	} FILAS
2	51	27	12	5	3	
3	14	32	7	24	5	
	} COLUMNAS					

- ☀ Existen conjuntos de datos que por su naturaleza se representan mejor mediante matrices multidimensionales que por vectores.
- ☀ Por ejemplo, se puede representar las calificaciones de 4 asignaturas cursadas por 6 estudiantes mediante una matriz

Definición

- ⚙ Una **matriz** se define utilizando **corchetes**.
- ⚙ Los **elementos** de cada **columna** se separan mediante **espacios** y las **diferentes filas** por **punto y coma**.

⚙ EJEMPLO:

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

A =

1	3	5
6	9	2
4	8	7

Definición

- ☀ Una **matriz** puede ser **definida** mediante la **composición** de otras matrices. Para ello:
 - ☀ Se utilizarán **corchetes**.
 - ☀ Si escribimos un **espacio** entre las matrices, la **composición** será por **filas**.
 - ☀ Si escribimos un **punto y coma** entre ellas, la **composición** será por **columnas**.

Definición

⚙ EJEMPLO:

```
>> A = [1 3;2 6];
```

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

```
B =
```

1	3	1	3
2	6	2	6

```
>> C = [A ; A]
```

```
C =
```

1	3
2	6
1	3
2	6

Acceso a los elementos de una matriz

- ☀ Para acceder a los elementos de una matriz se utilizan los paréntesis.

☀ EJEMPLO:

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

```
A =
```

1	2	3
4	5	6

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

```
ans =
```

```
5
```

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

```
ans =
```

2	3
5	6

Álgebra matricial

- ⚙ Las operaciones **suma** y **resta** operan **elemento a elemento**.
- ⚙ La operación **producto** es el **producto matricial**
- ⚙ Existen dos operaciones división :
 - ⚙ **A/b** es la solución al sistema $xA = b$, es decir, algo así como $b * \text{inv}(A)$
 - ⚙ **$A \backslash b$** es la solución al sistema $Ax = b$, es decir, algo así como $\text{inv}(A) * b$

Álgebra matricial

- ☀ La operación $^$ es la **potencia matricial**, por lo que exige que la **matriz sea cuadrada**.
- ☀ La **multiplicación, división y potencia punto a punto** se expresan con $.^*$, $./$ y $.^$ respectivamente.
- ☀ Las **funciones trigonométricas** (\sin , \cos , ...) y **exponenciales** (\exp , \log , ...) operan **elemento a elemento**.

Álgebra matricial

⚙ EJEMPLOS:

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

```
A =
```

```
    1    2  
    3    4
```

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

```
B =
```

```
    1  
    2
```

```
C = A * B
```

```
C =
```

```
    5  
   11
```

```
>> D = C .* B
```

```
D =
```

```
    5  
   22
```

```
>> E = A \ B
```

```
E =
```

```
    0  
  0.5000
```

```
>> F = A/B
```

```
??? Error using ==> mrdivide  
Matrix dimensions must agree.
```

Operaciones sobre matrices

☀ Funciones para rellenar una matriz:

FUNCIÓN	SIGNIFICADO
<code>zeros()</code>	Rellena con ceros la matriz
<code>ones()</code>	Rellena con unos la matriz
<code>rand()</code>	Rellena con valores aleatorios según una distribución uniforme
<code>randn()</code>	Rellena con valores aleatorios según una distribución gaussiana
<code>linspace()</code>	Valores equiespaciados (devuelve un vector)
<code>logspace()</code>	Valores equiespaciados de forma logarítmica
<code>eye()</code>	Matriz identidad
<code>tril()</code>	Extrae la matriz triangular inferior
<code>triu()</code>	Extrae la matriz triangular superior

Operaciones sobre matrices

☀ Funciones más usuales:

FUNCIÓN	SIGNIFICADO
det()	Calcula el determinante de una matriz
inv()	Calcula la inversa de una matriz
rank()	Devuelve el rango de una matriz
expm()	Exponencial matricial
eig()	Valores y vectores propios de una matriz
size()	Devuelve el tamaño de una matriz (filas, columnas)

☀ Muchas funciones sobre vectores se pueden aplicar a las matrices (sum, prod, mean, sort, min, max, std). En estos casos, operan sobre las columnas, y devuelven un vector como resultado

Operaciones sobre matrices

⚙ EJEMPLOS:

```
>> A = zeros(2,3)
```

A =

0	0	0
0	0	0

```
>> B = ones(3,2)
```

B =

1	1
1	1
1	1

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

C =

0.8147	0.1270	0.6324
0.9058	0.9134	0.0975

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

D =

-0.4336	2.7694
0.3426	-1.3499
3.5784	3.0349

Operaciones sobre matrices

⚙ EJEMPLOS:

```
>> l = linspace(1,6,3)
```

```
l =
```

```
    1.0000    3.5000    6.0000
```

```
>> g = logspace(1,6,3)
```

```
g =
```

```
    1.0e+006 *
```

```
    0.0000    0.0032    1.0000
```

```
>> E = eye(3,2)
```

```
E =
```

```
    1    0
```

```
    0    1
```

```
    0    0
```

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

```
M =
```

```
    1    2    3
```

```
    4    5    6
```

```
    7    8    9
```

```
>> TI = tril(M)
```

```
TI =
```

```
    1    0    0
```

```
    4    5    0
```

```
    7    8    9
```


Operaciones sobre matrices

⚙ EJEMPLOS:

```
>> TS = triu(M)
```

```
TS =  
    1    2    3  
    0    5    6  
    0    0    9
```

```
>> D = det(M)
```

```
D =  
    0
```

```
>> I = inv(M)
```

```
1.0e+016 *  
   -0.4504   0.9007   -0.4504  
    0.9007  -1.8014   0.9007  
   -0.4504   0.9007   -0.4504
```

```
>> R = rank(M)
```

```
R =  
    2
```

```
>> E = expm(M)
```

```
E =  
1.0e+006 *  
   1.1189   1.3748   1.6307  
   2.5339   3.1134   3.6929  
   3.9489   4.8520   5.7552
```

```
>> V = eig(M)
```

```
16.1168  
-1.1168  
0.0000
```

Operaciones sobre matrices

⚙ EJEMPLOS:

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

```
A =
```

```
     2     3     4
     6     7     8
```

```
>> t = size(A)
```

```
t =
```

```
     2     3
```

Estructuras de datos en Matlab

- ☀ Matrices: Se accede a los datos por índices y todos son del mismo tipo.
- ☀ Celdas: Se accede a los datos por índices y no todos son del mismo tipo.
- ☀ Registros: Se accede a los datos por nombre y no todos son del mismo tipo.
- ☀ Y ficheros...

Matrices de celdas

⚙ EJEMPLOS:

```
» clear; A={'stuff' [1 2;3 4] [0:4]}; whos
Name      Size      Bytes  Class
A         1x3         358   cell array
Grand total is 17 elements using 358 bytes
```

```
» A
A =
    'stuff'    [2x2 double]    [1x5 double]
```

```
» A(2) % The parentheses provide a cell
ans =
    [2x2 double]
```

```
» A{2} % Curly brackets extract what's underneath
A =
     1     2
     3     4
```

Observar el uso de las llaves { }

Registros

⚙ EJEMPLOS:

```
» empleado.nombre = 'Lucas';  
» empleado.edad = 35;  
» empleado.hijos = {'Juan' 'Margarita' 'Lola'}
```

```
empleado =
```

```
    nombre : 'Lucas'  
      edad : 35  
     hijos : {1x3 cell}
```

```
» whos
```

Name	Size	Bytes	Class
empleado	1x1	692	struct array

Grand total is 25 elements using 692 bytes

Registros

- ☀ No hace falta crear el modelo de la estructura, pero si lo hacemos:

```
struct('campo1','valor','campo2','valor'....)    Solo comillas en campos y cadenas.
```

- ☀ Podemos añadirle un campo nuevo en cualquier momento.

```
estructura.nuevo=valor;
```

- ☀ Se pueden crear vectores de estructuras:

```
estructura(tam)=struct(....); %El valor se le asigna al último elemento
```

- ☀ Pueden crearse estructuras anidadas.

```
clase=struct('curso','primero','grupo','A', 'alum', struct('nombre','Juan', 'edad', 19))
```

Ficheros

⚙ Importar datos:

Copy&Paste

```
A=load('fichero.txt') %Carga el contenido en A
```

Se separan las columnas con espacio y las filas con intro.

⚙ Exportar datos:

Diary =>Para datos pequeños

Save -ascii 'fichero'

⚙ Lectura/Escritura

⚙ Abrir: [fi,te]=fopen('fich','c') Cerrar: st=fclose(fi)

⚙ fscanf/fprintf =>ficheros de texto

⚙ fread/fwrite =>ficheros binarios

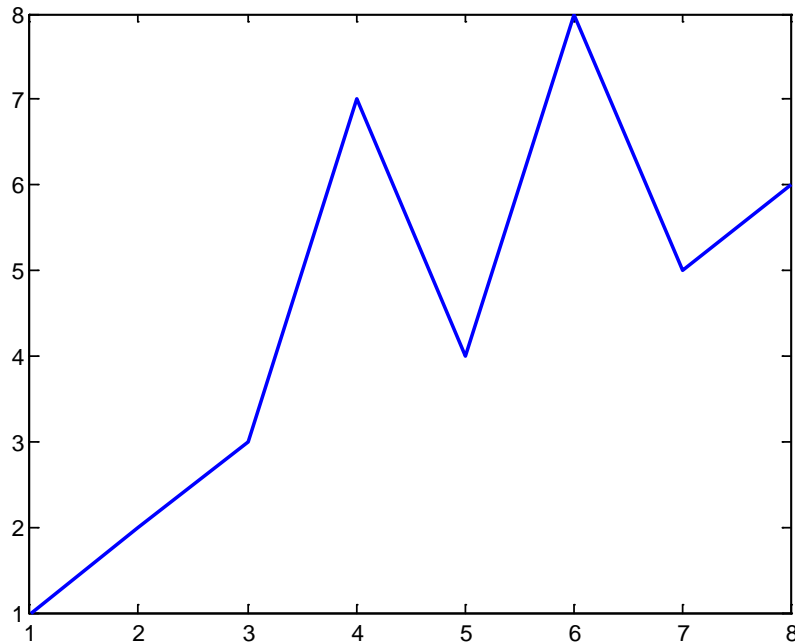
Gráficos en Matlab

Gráficos en Matlab

☀ La función plot() dibujar vectores.

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

```
>> plot(x)
```



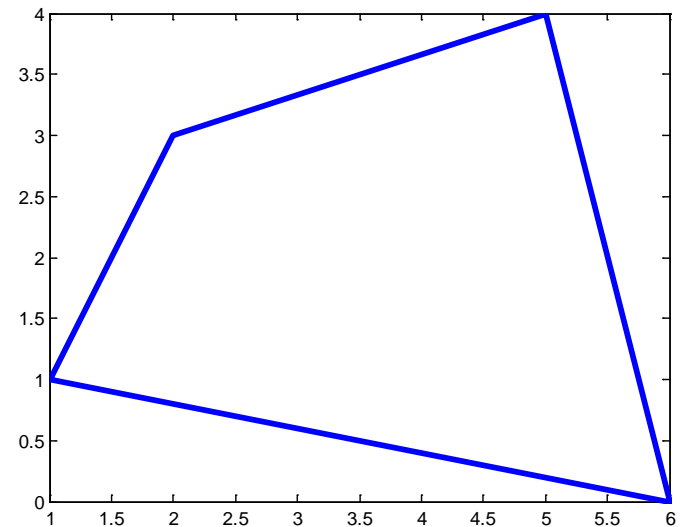
Gráficos en Matlab

- ☀ Una segunda forma de utilizar la función `plot()` es con dos vectores como argumentos. En este caso los elementos del segundo vector se representan en ordenadas frente a los valores del primero, que se representan en abscisas.

```
>> x=[1 6 5 2 1];
```

```
>> y=[1 0 4 3 1];
```

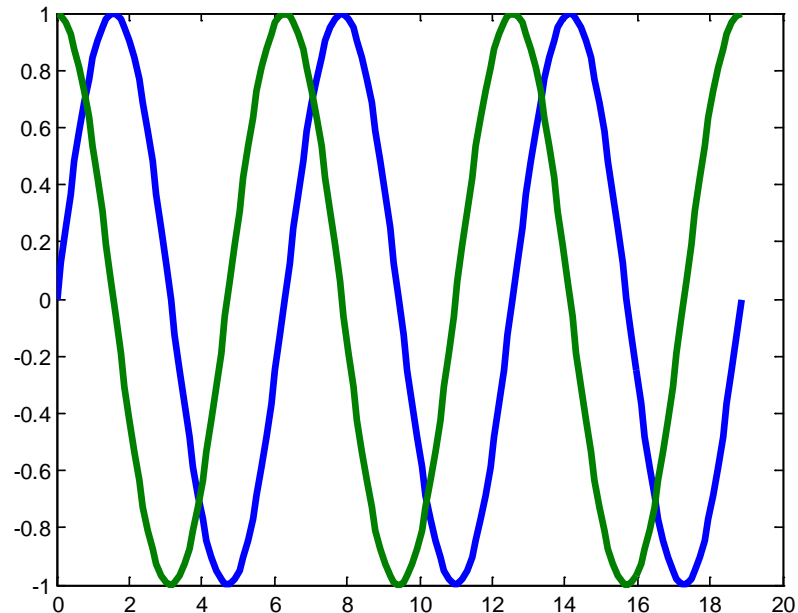
```
>> plot(x,y)
```



Gráficos en Matlab

- ☀ La función `plot()` permite también dibujar múltiples curvas introduciendo varias parejas de vectores como argumentos.

```
>> x=0:pi/25:6*pi;  
>> y=sin(x); z=cos(x);  
>> plot(x,y,x,z)
```



Estilos de líneas

Símbolo	Color	Símbolo	Marcadores (markers)
y	yellow	.	puntos
m	magenta	o	círculos
c	cyan	x	marcas en x
r	red	+	marcas en +
g	green	*	marcas en *
b	blue	s	marcas cuadradas (square)
w	white	d	marcas en diamante (diamond)
k	black	^	triángulo apuntando arriba
		v	triángulo apuntando abajo
Símbolo	Estilo de línea	>	triángulo apuntando a la dcha
-	líneas continuas	<	triángulo apuntando a la izda
:	líneas a puntos	p	estrella de 5 puntas
-.	líneas a barra-punto	h	estrella de seis puntas
--	líneas a trazos		

Gráficos en Matlab

☀ Marcas

```
>> x=[0 0.6 1.1 1.7 2.2 2.8 3.3 3.9 4.4 5]';
```

```
>> y=[0.5 0.7 0.9 1.2 1.5 2 2.6 3.5 4.6 6.1]';
```

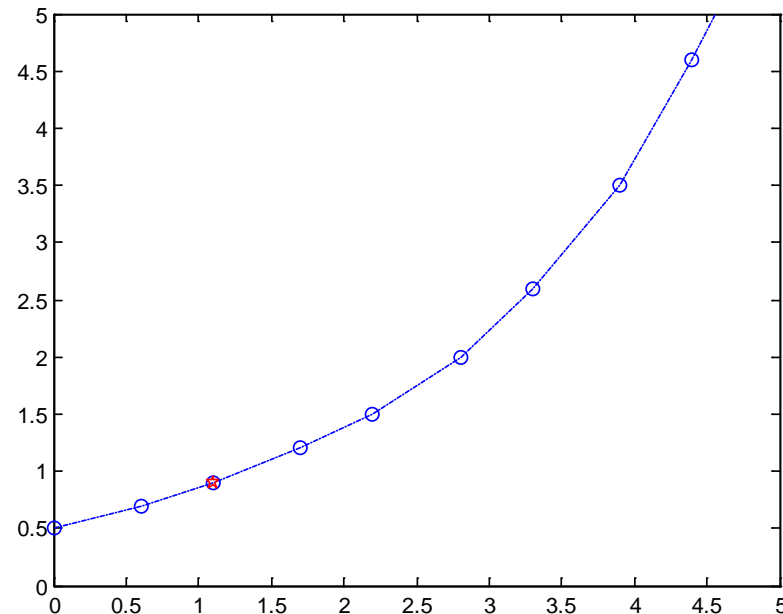
```
>> plot(x,y,'o')
```

```
>> axis([0 5 0 5])
```

```
>> hold on
```

```
>> plot(x,y,'-.'
```

```
>> plot(x(3),y(3),'pr')
```



Gráficos en Matlab

- ☀ Existe la posibilidad de añadir líneas a un gráfico ya existente, sin destruirlo o sin abrir una nueva ventana. Se utilizan para ello los comandos `hold on` y `hold off`. El primero de ellos hace que los gráficos sucesivos respeten los que ya se han dibujado en la figura (es posible que haya que modificar la escala de los ejes); el comando `hold off` deshace el efecto de `hold on`.

```
>> x1=[1 3 5 2 4];
```

```
>> x2=[3 4 2 6 7];
```

```
>> x3=[3 6 8 9 4];
```

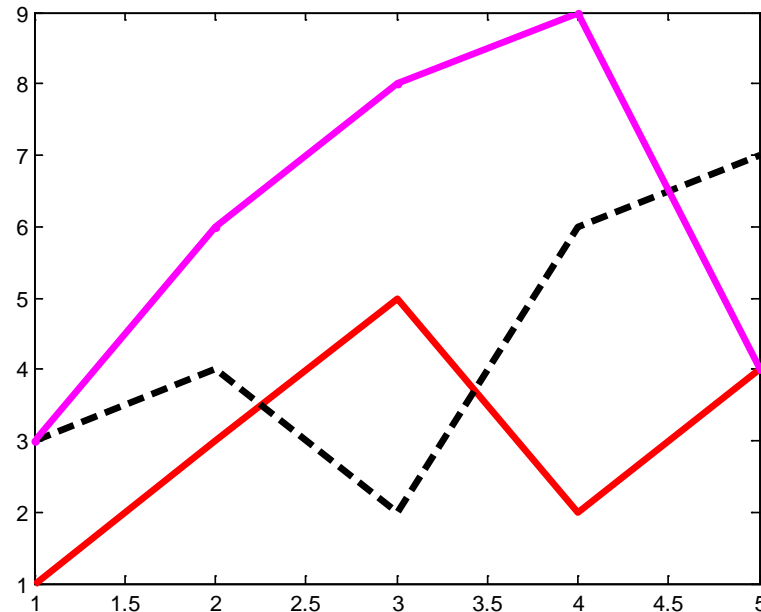
```
>> plot(x1,'r-')
```

```
>> hold on
```

```
>> plot(x2,'k--')
```

```
>> plot(x3,'m.-')
```

```
>> hold off
```



Gráficos en Matlab

☀ Una ventana gráfica se puede dividir en m particiones horizontales y n verticales, con objeto de representar múltiples gráficos en ella. Cada una de estas subventanas tiene sus propios ejes, aunque otras propiedades son comunes a toda la figura. La forma general de este comando es:

```
>> subplot(m,n,i)
```

donde m y n son el número de subdivisiones en filas y columnas, e i es la subdivisión que se convierte en activa.

Subplot

```
x=1:0.1:100
```

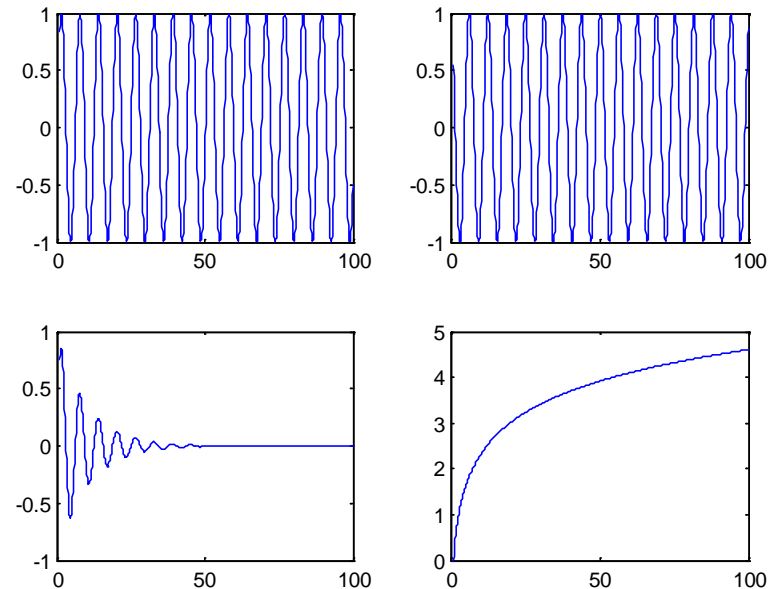
```
y=sin(x); z=cos(x); w=exp(-x*.1).*y; v=log(x);
```

```
subplot(2,2,1), plot(x,y)
```

```
subplot(2,2,2), plot(x,z)
```

```
subplot(2,2,3), plot(x,w)
```

```
subplot(2,2,4), plot(x,v)
```



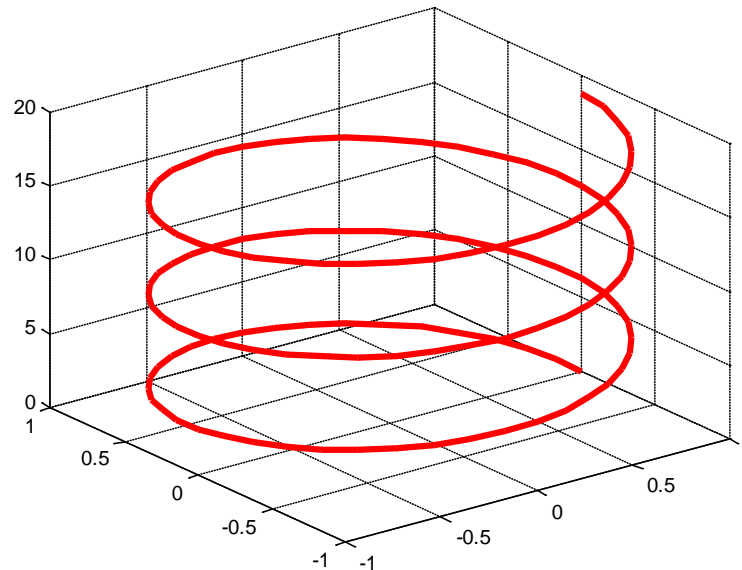
Gráficos en Matlab

- ☀ Si se llama a la función `figure` sin argumentos, se crea una nueva ventana gráfica con el número consecutivo que le corresponda. El valor de retorno es dicho número.
- ☀ El comando `figure(n)` hace que la ventana `n` pase a ser la ventana o figura activa. Si dicha ventana no existe, se crea una nueva ventana con el número consecutivo que le corresponda
- ☀ La función `close` cierra la figura activa, mientras que `close(n)` cierra la ventana o figura número `n`.
- ☀ El comando `clf` elimina el contenido de la figura activa, es decir, la deja abierta pero vacía.
- ☀ La función `gcf` devuelve el número de la figura activa en ese momento.
- ☀ `xlabel`, `ylabel`, `title`

Gráficos tridimensionales

- ☀ La primera forma de gráfico 3D es la función `plot3`, que es el análogo tridimensional de la función `plot`.

```
>> fi=[0:pi/20:6*pi]; plot3(cos(fi),sin(fi),fi,'r'), grid
```



Gráficos tridimensionales

- ☀ Dibujar una *función de dos variables* ($z=f(x,y)$) sobre un dominio rectangular.
- ☀ 1- Definir x e y , dos vectores que contienen las coordenadas en una y otra dirección de la retícula (*grid*) sobre la que se va a dibujar la función.
- ☀ 2- Después hay que crear dos matrices X (cuyas filas son copias de x) e Y (cuyas columnas son copias de y).
- ☀ 3- Estas matrices se crean con la función **meshgrid**. Estas matrices representan respectivamente las coordenadas x e y de todos los puntos de la retícula.
- ☀ Finalmente hay que dibujar esta matriz Z con la función **mesh**, cuyos elementos son función elemento a elemento de los elementos de X e Y .

Gráficos tridimensionales

```
close all
```

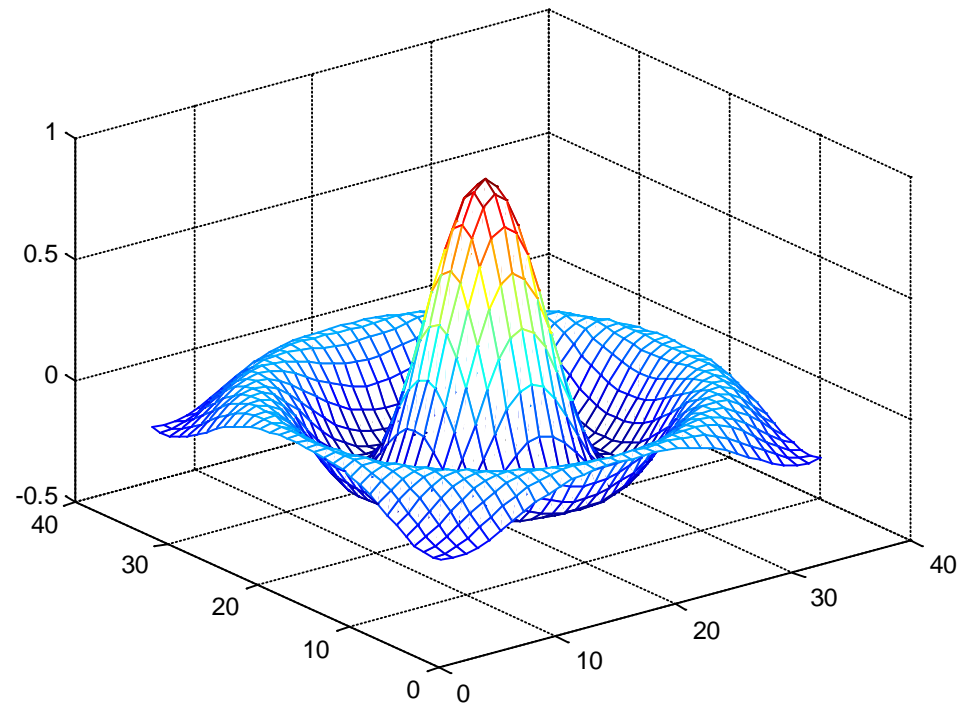
```
u=-8:0.5:8; v=u;
```

```
[U,V]=meshgrid(u,v);
```

```
R=sqrt(U.^2+V.^2)+eps;
```

```
W=sin(R)./R;
```

```
mesh(W)
```



Gráficos tridimensionales

```
x=[-3:0.4:3]; y=x;
```

```
close
```

```
subplot(2,2,1), figure(gcf), fi=[0:pi/20:6*pi];
```

```
plot3(cos(fi),sin(fi),fi,'r'), grid
```

```
[X,Y]=meshgrid(x,y);
```

```
Z= 3*(1-X).^2.*exp(-(X.^2) - (Y+1).^2) ...
```

```
- 10*(X/5 - X.^3 - Y.^5).*exp(-X.^2-Y.^2) ...
```

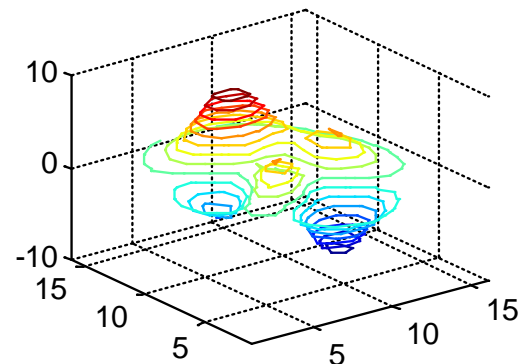
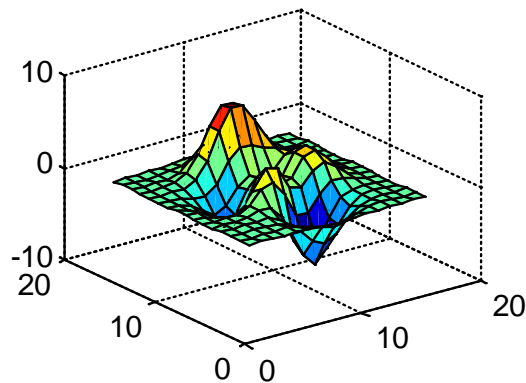
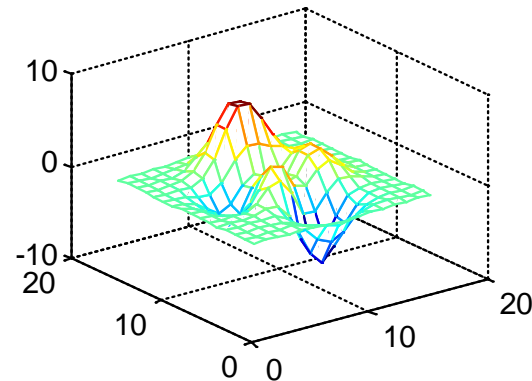
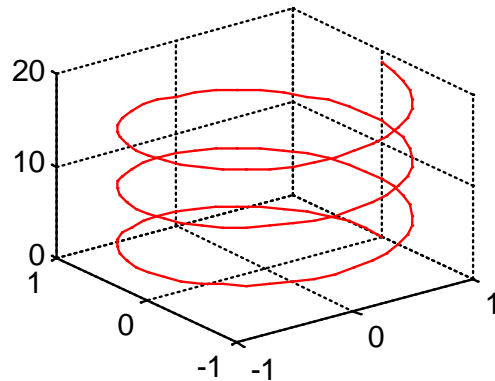
```
- 1/3*exp(-(X+1).^2 - Y.^2);
```

```
subplot(2,2,2), figure(gcf), mesh(Z)
```

```
subplot(2,2,3), figure(gcf), surf(Z)
```

```
subplot(2,2,4), figure(gcf), contour3(Z,16)
```

Gráficos tridimensionales



Tratamiento de imágenes en Matlab

Tratamiento de imágenes

`A = imread(FILENAME,FMT)`

Lee una imagen de un fichero ('Filename') que se encuentra en el directorio de trabajo. Se puede utilizar la ruta donde se encuentra el fichero.

'Fmt' Es el formato del fichero según la extensión.

A es una matriz de NxM si es una imagen en escala de grises o una matriz NxMx3 si es en color.

Ej: `A = imread('ngc6543a.jpg');`

=>Hacer `help imread`

Tratamiento de imágenes

`imwrite(A,FILENAME,FMT)`

Graba la imagen almacenada en la matriz A al fichero especificado por 'Filename' en el formato especificado por 'fmt'.

Ej: `A = rand(50); imwrite(A, 'prueba.jpg')`

=>Hacer `help imwrite`

Tratamiento de imágenes

`imshow` => Muestra imagen

`imagesc` => Muestra de imagen con la escala

`imresize` => Redimensiona imágenes

`imtool` => Información sobre el fichero de imágenes

`im2bw` => Convierte a blanco/negro

`rgb2gray` => De rgb a escala de grises

`frame2im` => Convierte de un frame de película a imagen

`Im2frame` => Convierte una imagen en un frame

`Open / close` => Abre o cierra un fichero

`Videowriter` => Crea un fichero de video

`Writevideo` => Guarda una imagen/frame a video

Tratamiento de imágenes

```
v = VideoWriter('Pruebavideo.avi');  
  
open(v);  
  
Z = peaks; // Peaks es una demo de matlab  
  
surf(Z);  
  
axis tight manual  
  
set(gca,'nextplot','replacechildren');  
  
for k = 1:20  
  
    surf(sin(2*pi*k/20)*Z,Z)  
  
    frame = getframe(gcf);  
  
    writeVideo(v,frame);  
  
end  
  
close(v);
```

Tratamiento de imágenes

```
v = VideoReader('xylophone.mpg');  
numFrames = get(v, 'NumberOfFrames');  
A=read(v);  
for i=1:numFrames  
    imshow(A(:,:,:,i),[]);  
    pause;  
end;
```

Webcam

```
figure;  
cam = webcam(1);  
set(gcf,'currentchar',' ');  
while get(gcf,'currentchar')== ' ',  
    img = snapshot(cam);  
    datagrey = rgb2gray(img);  
    imshow(datagrey,[]);  
end  
close  
clear cam
```

Algunos conceptos de preprocesamiento de imágenes

- ☀ Histograma: Representación gráfica de las frecuencias de grises en una imagen que proporciona información estadística de la imagen

Ej:

```
imagen = imread('cameraman.tif');  
imhist(imagen)
```

	0	3	3	2	5	5	
	1	1	0	3	4	5	
	2	2	2	4	4	4	
	3	3	4	4	5	5	
	3	4	5	5	6	6	
	7	6	6	6	6	5	

$$H(g) = \frac{N_g}{N_{total}}$$

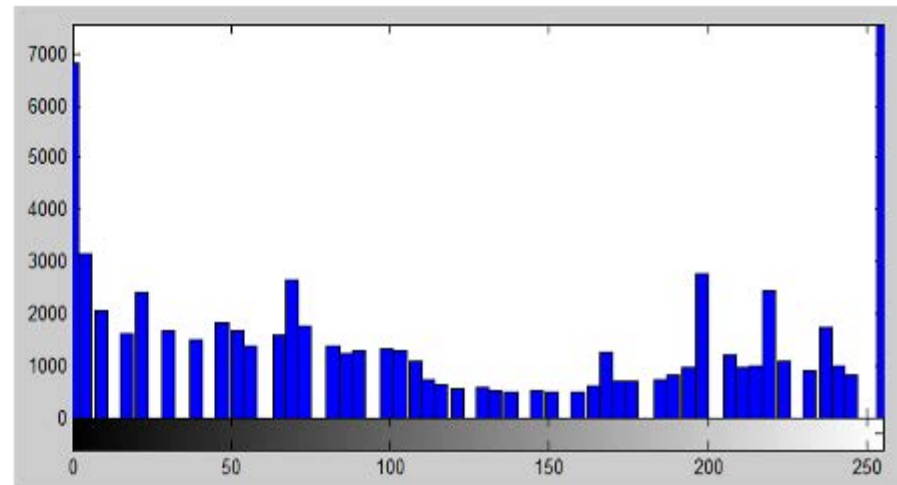
Gray Value	Count	Rel. Freq.
0	2	.05
1	2	.05
2	4	.11
3	6	.17
4	7	.20
5	8	.22
6	6	.17
7	1	.03

Algunos conceptos de preprocesamiento de imágenes

- ☀ Brillo: valor medio de la imagen (del histograma).
- ☀ Contraste: Dispersión de los niveles de gris (relacionado con la varianza, ya que, a mayor varianza, mayor contraste).
- ☀ Energía: Indica el grado de dispersión de los niveles de gris (valor máximo cuando sólo hay un nivel de gris).
- ☀ Entropía: Indica el desorden en la imagen, es decir, a más entropía, más niveles de grises participan en la imagen.

Algunos conceptos de preprocesamiento de imágenes

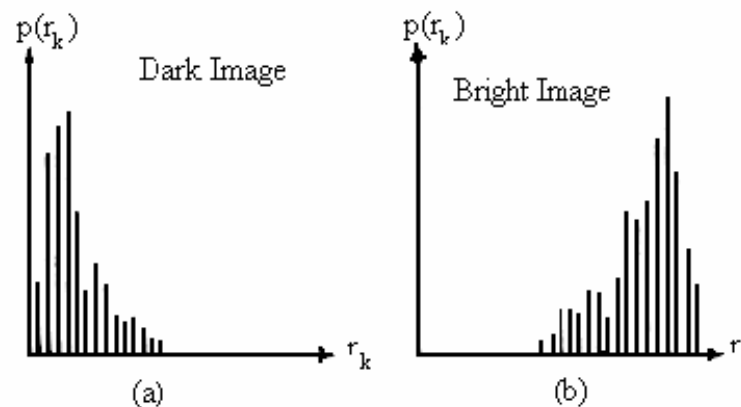
- ☀ Saturación: El histograma presenta valores muy altos en sus extremos del rango dinámico (forma de U).



Algunos conceptos de preprocesamiento de imágenes

☀ Brillo:

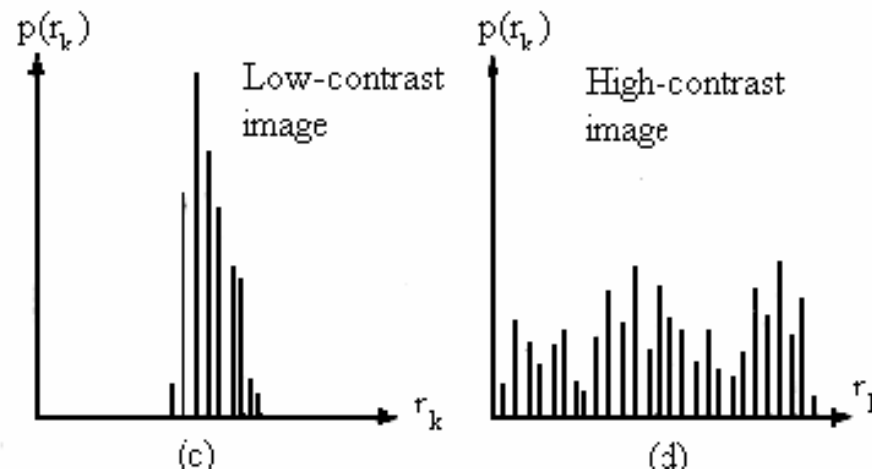
- ☀ Imagen oscura (niveles de gris concentrados en la zona baja del rango)
- ☀ Imagen brillante (niveles de gris concentrados en la parte alta del rango)



Algunos conceptos de preprocesamiento de imágenes

☀ Contraste:

- ☀ Contraste bajo (distribución de niveles de gris concentrados en una determinada zona)
- ☀ Contraste alto (amplia distribución de los niveles de gris)

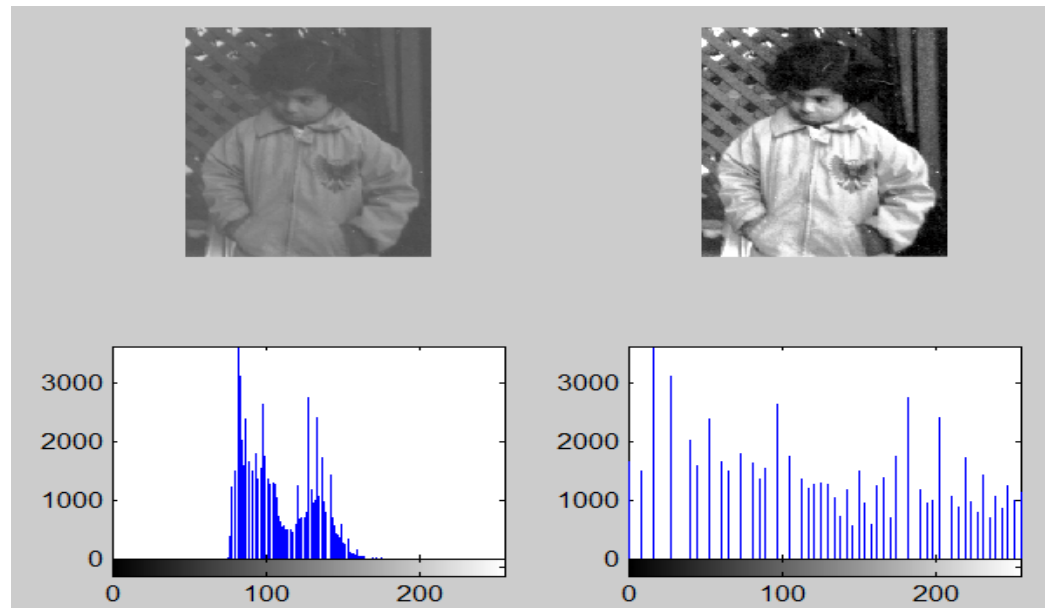


Algunos conceptos de preprocesamiento de imágenes

- ☀ Modificar el contraste (Ecualizar el histograma):
 - ☀ Expande la distribución de los niveles de gris para mejorar el contraste de una imagen.
 - ☀ Comprime píxeles en las zonas planas del histograma sobre un conjunto pequeño de niveles de grises.
 - ☀ Aplana el histograma.

Algunos conceptos de preprocesamiento de imágenes

```
imagen = imread('pout.tif');  
subplot(2,2,1);  
imshow(imagen);  
subplot(2,2,3);  
imhist(imagen), axis tight;  
subplot(2,2,2);  
imagen_eq = histeq(imagen);  
imshow(imagen_eq);  
subplot(2,2,4);  
imhist(imagen_eq), axis tight;
```



Algunos conceptos de preprocesamiento de imágenes

- ☀ Suavizado: reducción del ruido y/o artefactos que pueden presentarse en una imagen a consecuencia del proceso de captura, digitalización y/o transmisión.
- ☀ Filtros lineales: Convolución de una imagen con una máscara predefinida
- ☀ Filtros no lineales: operación con los píxeles del entorno de vecindad

Algunos conceptos de preprocesamiento de imágenes

☀ Filtros lineales:

- ☀ Filtro de la media: El valor de cada píxel se obtiene promediando los valores de los píxeles en un entorno de vecindad predefinido (mascara).
- ☀ Filtro gaussiano: El valor de cada pixel es el resultado de promediar con distintos pesos los valores vecinos a ambos lados del pixel.

☀ Filtros no lineales:

- ☀ Filtro de la mediana: Un pixel se genera calculando la mediana del conjunto de píxeles del entorno de vecindad

Algunos conceptos de preprocesamiento de imágenes

☀ Filtros lineales en Matlab => Inconvenientes: Emborronamiento de los bordes (usar con precaución). El tamaño de la mascara es importante

☀ `imfilter` => Aplica el filtro

☀ `fspecial` => Genera la mascara

☀ `imnoise` => Añade ruido

☀ Filtros no lineales en Matlab

☀ `medfilt2` => Filtro de la mediana

(hacer help de las funciones anteriores)