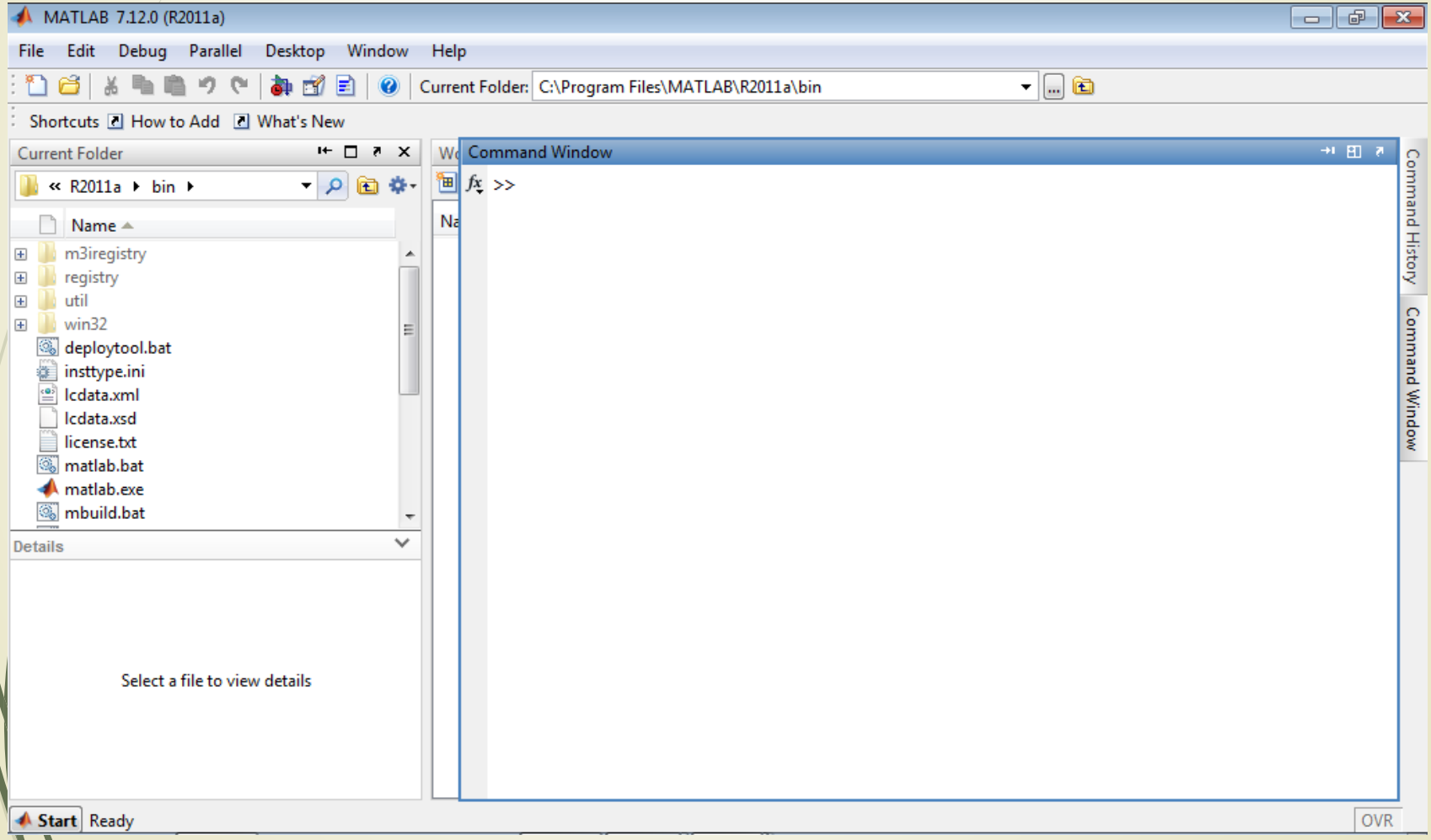
Computación

Introducción. Variables. Matrices. Funciones.

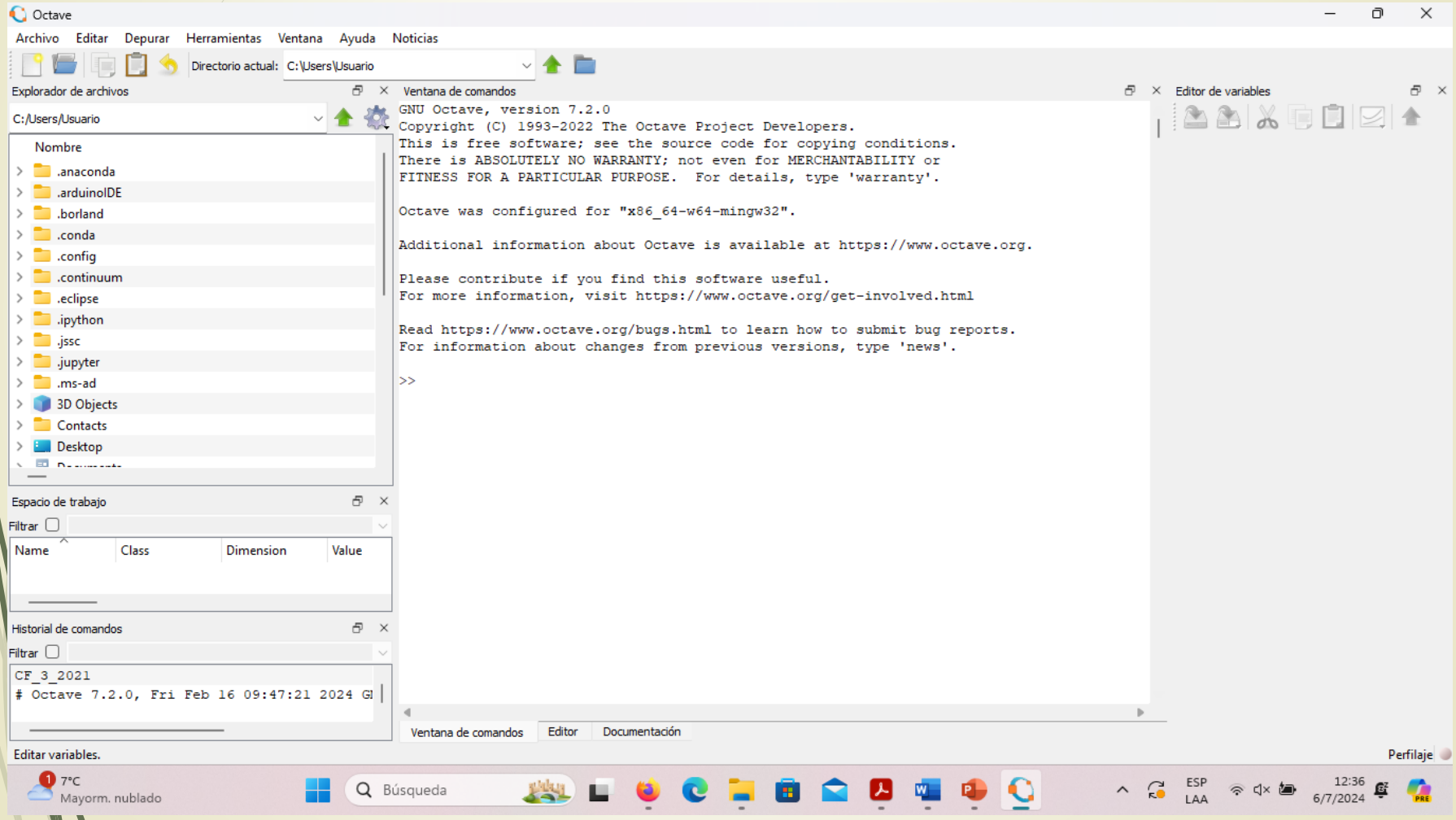
Estructuras de control.

Curso 2024

Escritorio en Matlab



Escritorio en Octave



Escritorio en Matlab

- **Command Windows**: Donde se ejecutan todas las instrucciones y programas. Se escribe la instrucción o el nombre del programa y Enter.

- **Command History**: Muestra los últimos comandos ejecutados en Command Windows. Se puede recuperar el comando haciendo doble

- **Current directory**: Situarse en el directorio donde se va a trabajar

Escritorio en Matlab

- **Help** (también se puede usar desde comand windows)
- **Workspace (espacio de trabajo)**: Para ver las variables que se están usando y sus dimensiones (si son matrices)
- **Editor del Matlab**: Todos los ficheros de comandos Matlab deben de llevar la extensión .m

- Espacio de trabajo: es el conjunto de variables que en un momento dado están definidas en la memoria del MATLAB. Las variables creadas desde la línea de comandos de MATLAB pertenecen al base workspace (espacio de trabajo base). Los mismo sucede con las variables creadas por scripts que se ejecutan desde la línea de comandos. Estas variables permanecen en el base workspace cuando se termina la ejecución del script y se mantienen allí durante toda la sesión de trabajo o hasta que se borren.

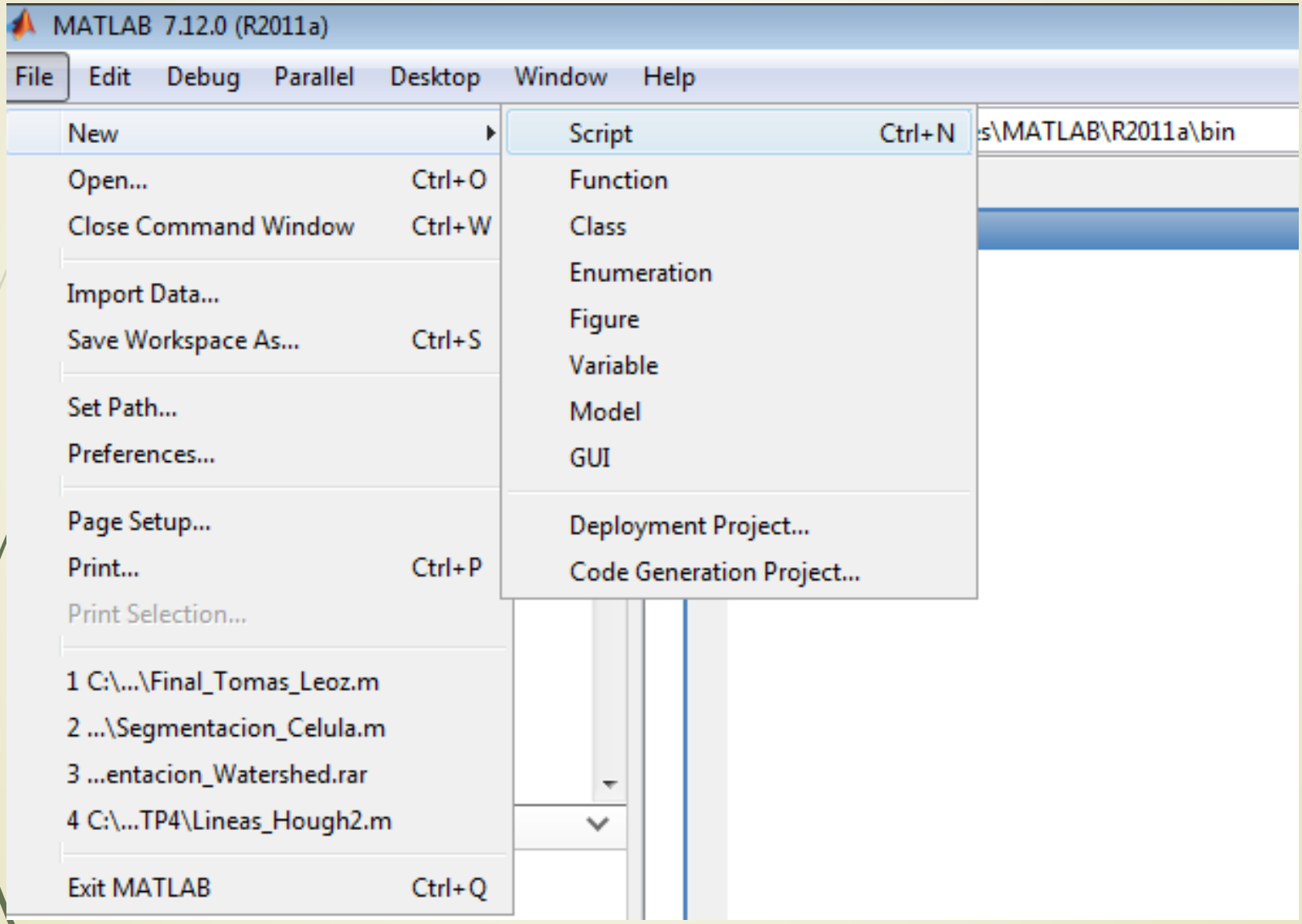
- Sin embargo, las variables creadas por una función pertenecen al espacio de trabajo de dicha función, que es independiente del espacio de trabajo base. Es decir, las variables de las funciones son LOCALES: MATLAB/OCTAVE reserva una zona de memoria cuando comienza a ejecutar una función, almacena en esa zona las variables creadas por esa función, y "borra" dicha zona cuando termina la ejecución de la función. Para hacer que una variable de una función pertenezca al base workspace, hay que declararla GLOBAL (global variable).

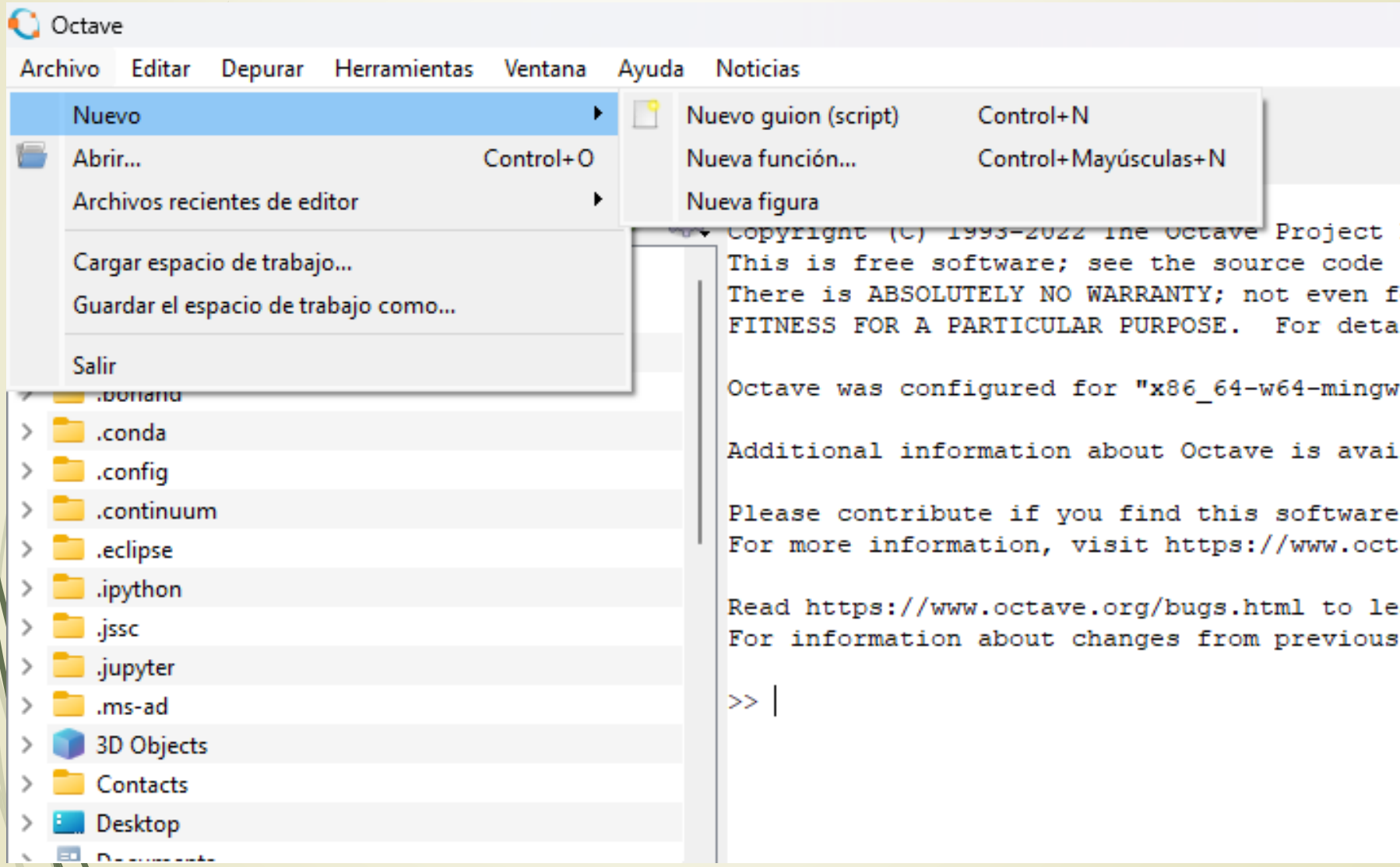
Script

- Un script es un conjunto de instrucciones (de cualquier lenguaje) guardadas en un archivo (usualmente de texto) que son ejecutadas normalmente mediante un intérprete.
- Son útiles para automatizar pequeñas tareas. También puede hacer las veces de un "programa principal" para ejecutar una aplicación.
- Así, para llevar a cabo una tarea, en vez de escribir las instrucciones una por una en la línea de comandos de MATLAB/OCTAVE, se pueden escribir una detrás de otra en un archivo.

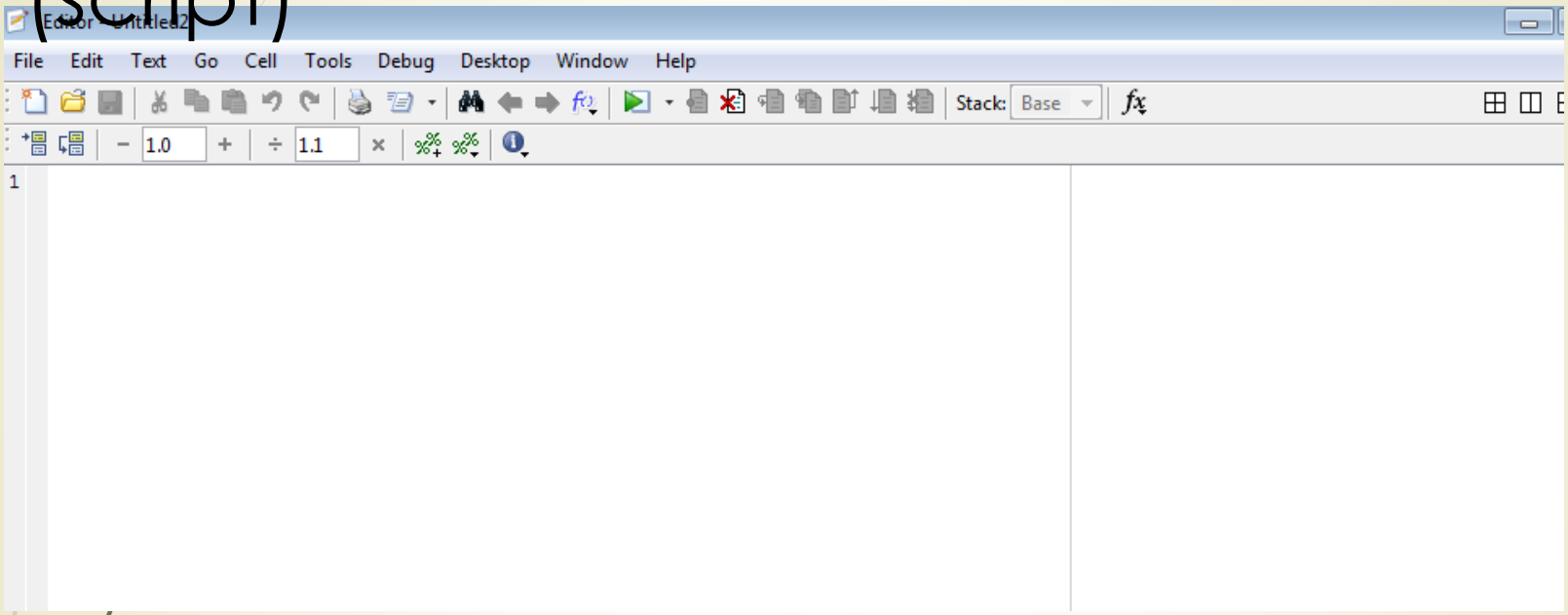
Script

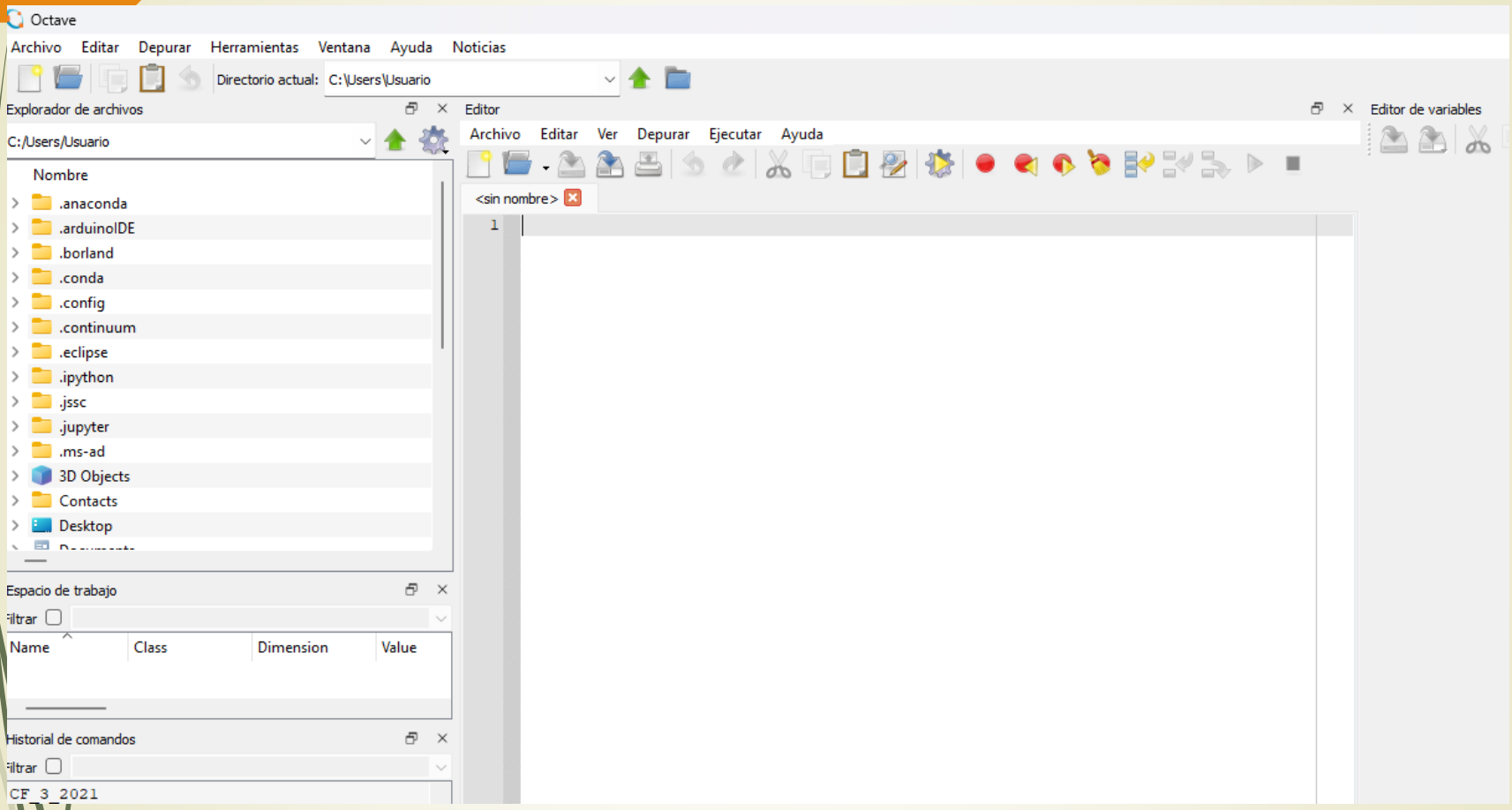
- Para ello se puede utilizar el Editor integrado: icono “hoja en blanco” del menú de herramientas, opción “. New M-file ” del Menú “ File”. Los scripts de MATLAB/OCTAVE deben guardarse en un archivo .m.
- Para ejecutar un script que esté en el directorio de trabajo, basta escribir su nombre en la línea de comandos


MATLAB 7.12.0 (R2011a)
File
Edit
Debug
Parallel
Desktop
Window
Help
New
Open... Ctrl+O
Close Command Window Ctrl+W
Import Data...
Save Workspace As... Ctrl+S
Set Path...
Preferences...
Page Setup...
Print... Ctrl+P
Print Selection...
1 C:\...\Final_Tomas_Leoz.m
2 ...\Segmentacion_Celula.m
3 ...entacion_Watershed.rar
4 C:\...TP4\Lineas_Hough2.m
Exit MATLAB Ctrl+Q
Script Ctrl+N
Function
Class
Enumeration
Figure
Variable
Model
GUI
Deployment Project...
Code Generation Project...
s\MATLAB\R2011a\bin


Octave
Archivo
Editar
Depurar
Herramientas
Ventana
Ayuda
Noticias
Nuevo
Abrir... Control+O
Archivos recientes de editor
Cargar espacio de trabajo...
Guardar el espacio de trabajo como...
Salir
Nuevo guion (script) Control+N
Nueva función... Control+Mayúsculas+N
Nueva figura
Copyright (C) 1993-2022 The Octave Project
This is free software; see the source code
There is ABSOLUTELY NO WARRANTY; not even f
FITNESS FOR A PARTICULAR PURPOSE. For deta
Octave was configured for "x86_64-w64-mingw
Additional information about Octave is avai
Please contribute if you find this software
For more information, visit https://www.oct
Read https://www.octave.org/bugs.html to le
For information about changes from previous
>>
.conda
.config
.continuum
.eclipse
.ipython
.jssc
.jupyter
.ms-ad
3D Objects
Contacts
Desktop

Editor para generar el archivo .m (script)




Octave
Archivo Editar Depurar Herramientas Ventana Ayuda Noticias
Directorio actual: C:\Users\Usuario
Explorador de archivos
C:/Users/Usuario
Nombre
.anaconda
.arduinoIDE
.borland
.conda
.config
.continuum
.eclipse
.ipython
.jssc
.jupyter
.ms-ad
3D Objects
Contacts
Desktop
Editor
Archivo Editar Ver Depurar Ejecutar Ayuda
<sin nombre>
1
Editor de variables
Espacio de trabajo
Filtrar
Name Class Dimension Value
Historial de comandos
Filtrar
CF_3_2021

Variables

- Las variables no son nunca declaradas: su tipo y su tamaño cambian de forma dinámica de acuerdo con los valores que le son asignados.
- Así, una misma variable puede ser utilizada, por ejemplo, para almacenar un número complejo a continuación, una matriz 25x40 de números enteros y luego para almacenar un texto.
- Las variables se crean automáticamente al asignarles un contenido.

Matrices y vectores

- Para definir una matriz no hace falta establecer de antemano su tamaño (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 punto y coma (;). Por ejemplo, el siguiente comando define una matriz A de dimensión (3x3):
- » A=[1 2 3; 4 5 6; 7 8 9]

Matrices y vectores

- Se accede a los elementos de un vector poniendo el índice entre paréntesis (por ejemplo x(3) ó x(i)). Los elementos de las matrices se acceden poniendo los dos índices entre paréntesis, separados por una coma (por ejemplo A(1,2) ó A(i,j)).

Definición de matrices

EJEMPLOS (construcciones elementales de matrices)

```
>> v=[1,-1,0,sin(2.88)]                    % vector fila longitud 4
>> w=[0;1.003;2;3;4;5*pi]                  % vector columna longitud 6
>> a=[1,2,3,4;5,6,7,8;9,10,11,12]          % matriz 3x4
>> mat=['Hola','Mari';'¿Como','estas?']    % matriz 2x2 de caracteres
```

Matrices

- Generación de una matriz de ceros, zeros(n,m)
- Generación de una matriz de unos, ones(n,m)
- Inicialización de una matriz identidad eye(n,m)
- Generación de una matriz de elementos aleatorios rand(n,m)
- Matrices con diagonal dada diag(v), diag(v,k)

Definición de vectores

- Vectores fila; elementos separados por blancos o comas

  >> v =[2 3 4]

- Vectores columna: elementos separados por punto y coma (;)

  >> w =[2;3;4;7;9;8]

- Dimensión de un vector w: length(w)
- Generación de vectores fila:
  - Especificando el incremento h de sus componentes v=a:h:b
  - Especificando su dimensión n: linspace(a,b,n) (por defecto n=100)

Definición de vectores

- Las siguientes funciones generan vectores de elementos regularmente espaciados, útiles por ejemplo para creación de gráficas.

`linspace(a,b,n)`	Si **a** y **b** son números reales y **n** un número entero, genera una partición regular del intervalo **[a,b]** con **n** nodos (**n**-1 subintervalos)
`linspace(a,b)`	Como el anterior, pero se asume **n**=100
`logspace(e,f,n)`	Vector con **n** elementos logarítmicamente espaciados desde 10^**e** hasta 10^**f**, es decir, cuyos logaritmos están regularmente espaciados. (`logspace(e,f,n) = 10.^linspace(e,f,n)`)
`logspace(e,f)`	Como el anterior, pero se asume **n**=50

Operaciones aritméticas

- Suma: +, Resta -
- Multiplicación: *, División: /
- Potencias: ^
- Orden de prioridad: Potencias, divisiones y multiplicaciones y por último sumas y restas. Usar () para cambiar la prioridad

Operaciones con escalares

- v+k adición o suma
- v-k sustracción o resta
- v*k multiplicación
- v/k divide por k cada elemento de v
- k./v divide k por cada elemento de v
- v.^k potenciación cada componente de v esta elevado a k
- k.^v potenciación k elevado cada componente de v

Operaciones entre matrices

- v+w adición o suma
- v-w sustracción o resta
- v.*w multiplicación cada elemento de v por el correspondiente de w
- v./w divide cada elemento de v por el correspondiente de w
- v.^w potenciación cada componente de v esta elevado al correspondiente de w
- Producto escalar de vectores v*w calcula el producto escalar de v por w

Scripts y funciones

- Los programas en Matlab se escriben como scripts o funciones.
- Scripts es un archivo .m que tiene una secuencia de sentencias en Matlab.
- Una función también es un archivo .m, pero es llamado desde un scripts (o también >>).
- Las variables definidas en la función, tienen alcance local (solo conocidas en esa función).

Funciones

- El usuario de Matlab/OCTAVE puede definir sus propias funciones o subrutinas y asignarle el nombre que quiera con la misma limitación que se tiene para nombrar un archivo.
- Esto es así porque de hecho definir una función propia consiste sencillamente en la creación de un archivo m que tiene por nombre el mismo nombre que el de la función.

Funciones

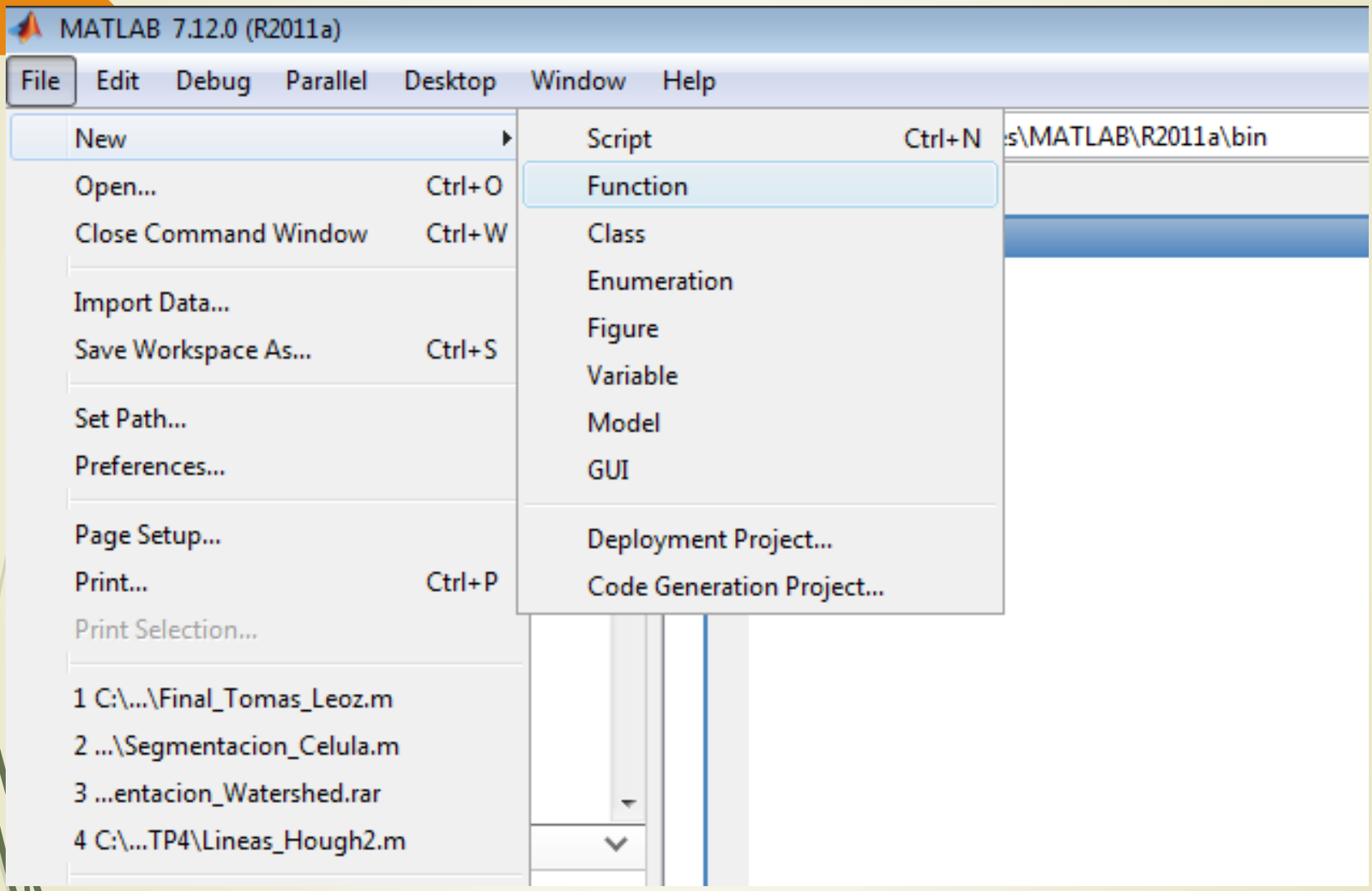
- Una función(habitualmente denominadas M-funciones en MATLAB), es un programa con una "interfase" de comunicación con el exterior mediante argumentos de entrada y de salida.
- Las funciones MATLAB responden al siguiente formato de escritura. La cláusula end del final no es obligatoria, excepto en el caso de funciones anidadas.

Funciones definidas por el programador

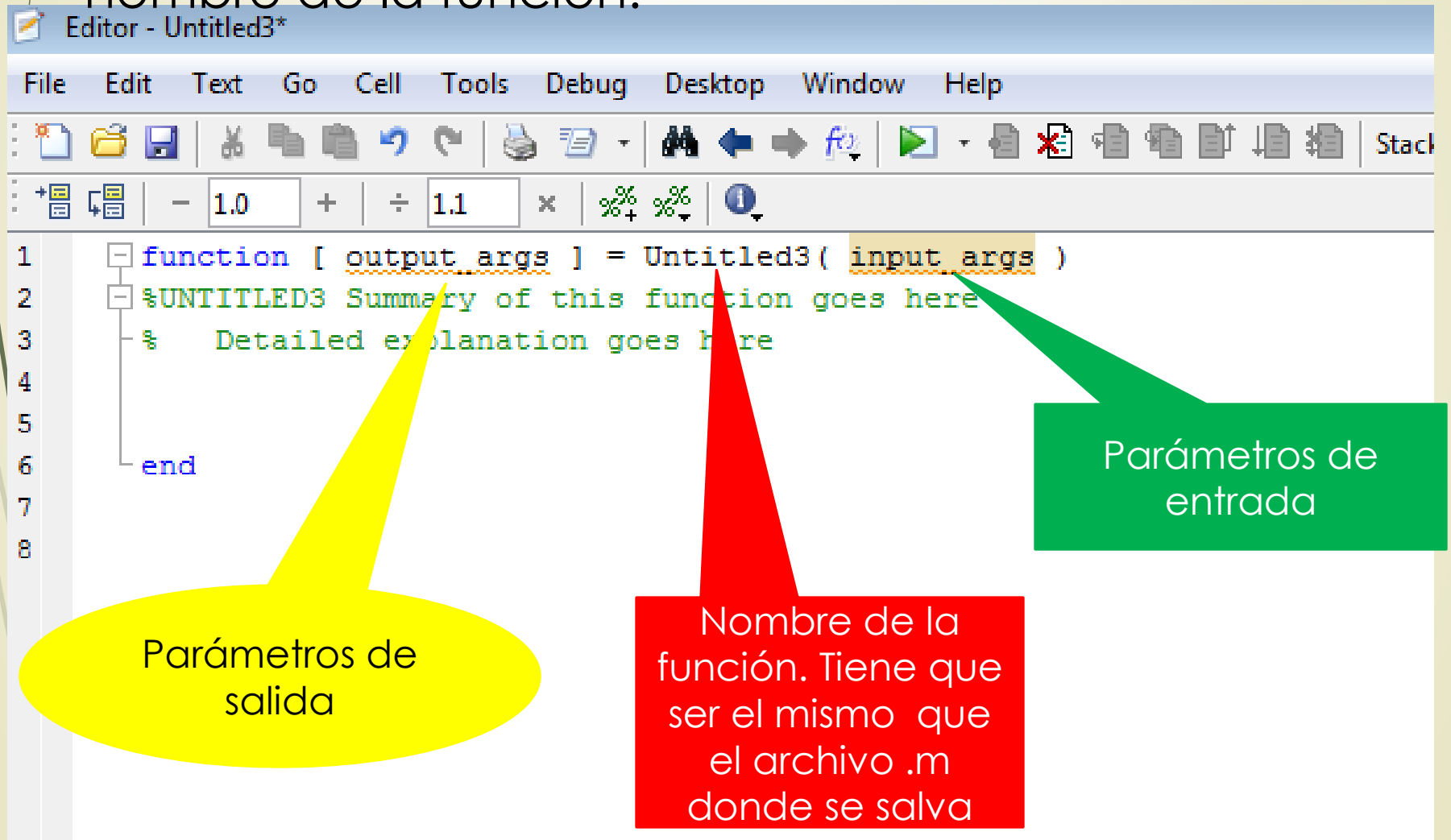
- La primera línea empieza con la palabra function y tiene la forma

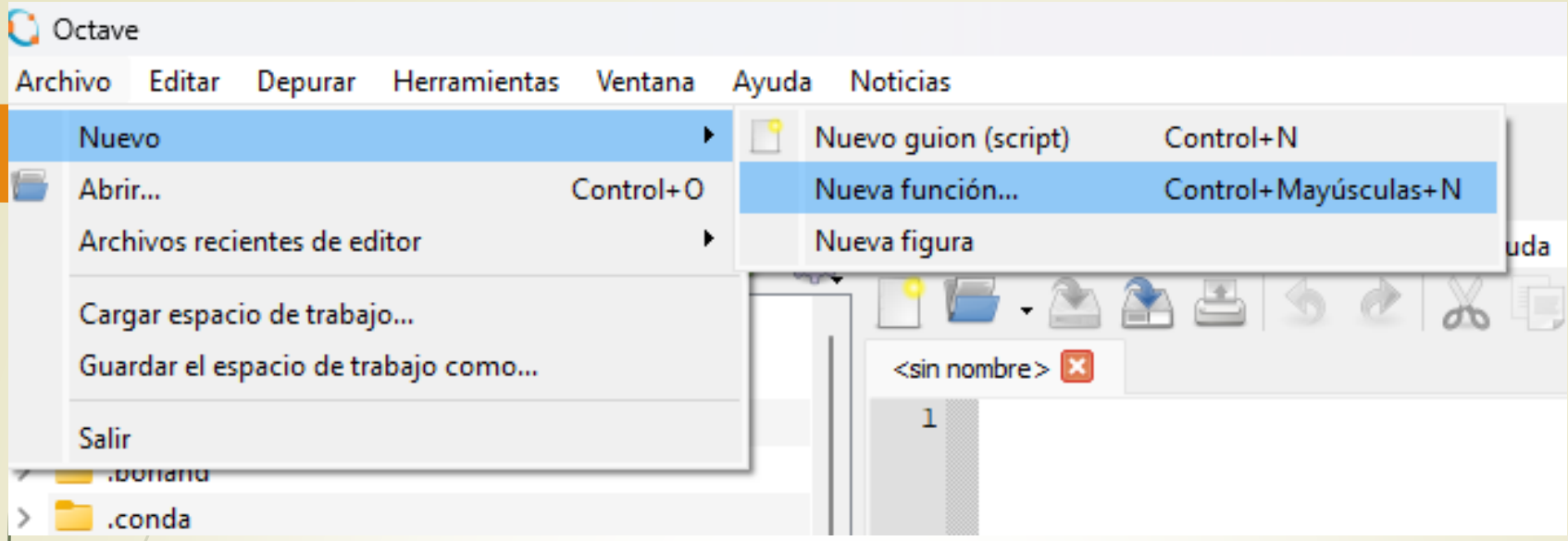
```
function
    [arg_salida]=nombre_funcion(arg_entrada
    )
function [out1, out2,…]=mifun(in1, in2, ….)
        sentencias
end
```

- El archivo se debe guardar con nombre mifun.m


MATLAB 7.12.0 (R2011a)
File
Edit
Debug
Parallel
Desktop
Window
Help
New
Open... Ctrl+O
Close Command Window Ctrl+W
Import Data...
Save Workspace As... Ctrl+S
Set Path...
Preferences...
Page Setup...
Print... Ctrl+P
Print Selection...
1 C:\...\Final_Tomas_Leoz.m
2 ...\Segmentacion_Celula.m
3 ...entacion_Watershed.rar
4 C:\...TP4\Lineas_Hough2.m
Script Ctrl+N
Function
Class
Enumeration
Figure
Variable
Model
GUI
Deployment Project...
Code Generation Project...
s\MATLAB\R2011a\bin

Editor para generar una función. Genera el esqueleto, hay que completar con las sentencias, argumentos de entrada y salida, nombre de la función.



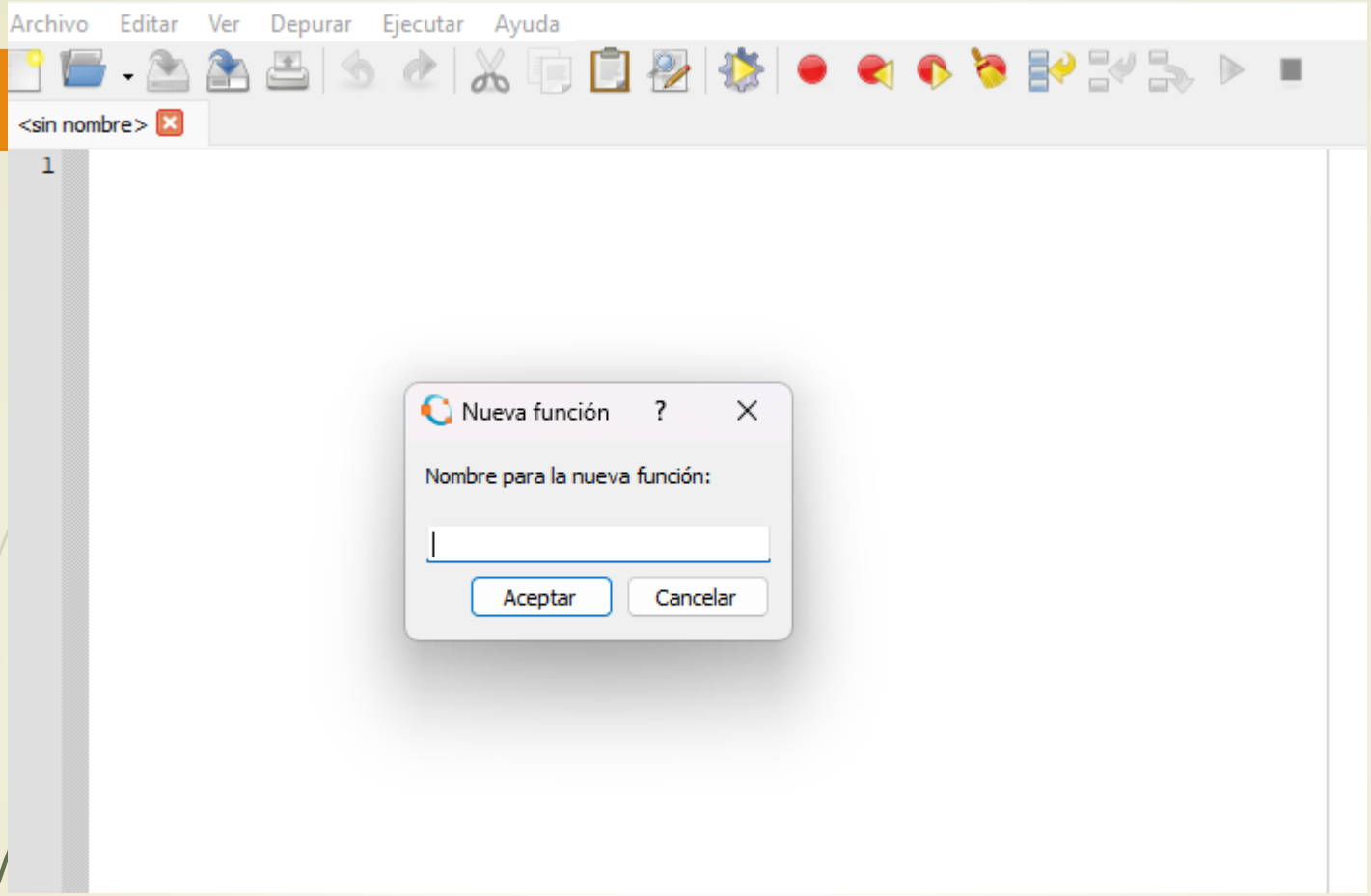



```
## Author: Usuario <Usuario@DESKTOP-8VEQM6P>
## Created: 2024-07-06

function retval = mia (input1, input2)

endfunction
```

a: 1 | Columna: 1 | Codificación: UTF-8 | Fin de línea: CRLF


Archivo
Editar
Ver
Depurar
Ejecutar
Ayuda
<sin nombre>
1
Nueva función
?
Nombre para la nueva función:
Aceptar
Cancelar

Funciones anónimas

- Las funciones anónimas constituyen una forma muy flexible de crear funciones "sobre la marcha", en la línea de comandos o bien en una línea cualquiera de una función o de un fichero .m. La forma general de las funciones anónimas es la siguiente:

```
>> nombre función=@(argumentos) expresión
```

Funciones anónimas: ejemplo

Supongamos la función: $y=a+bx+cx^2$

a=2; b=3; c=4

Definimos la función anónima siguiente:

```
>> f= @(x) a+b*x+c*x^2
>> f(0)
ans=2
```

Funciones elementales definidas

sqrt(x)	raiz cuadrada	sin(x)	seno (radianes)
abs(x)	módulo	cos(x)	coseno (radianes)
conj(z)	complejo conjugado	tan(z)	tangente (radianes)
real(z)	parte real	cotg(x)	cotangente (radianes)
imag(z)	parte imaginaria	asin(x)	arcoseno
exp(x)	exponencial	acos(x)	arcocoseno
log(x)	logaritmo natural	atan(x)	arcotangente
log10(x)	logaritmo decimal	cosh(x)	cos. hiperbólico
rat(x)	aprox. racional	sinh(x)	seno hiperbólico
mod(x,y) rem(x,y)	resto de dividir x por y . Iguales si x,y>0 . Ver help para definición exacta	tanh(x)	tangente hiperbólica
fix(x)	Redondeo hacia 0	acosh(x)	arcocoseno hiperb.
ceil(x)	Redondeo hacia + infinito	asinh(x)	arcoseno hiperb.
floor(x)	Redondeo hacia - infinito	atanh(x)	arcotangente hiperb.
round(x)	Redondeo al entero más próximo		

Funciones para graficar 2D

- plot() crea un gráfico a partir de vectores y/o columnas de matrices, con escalas lineales sobre ambos ejes.
- loglog() ídem con escala logarítmica en ambos ejes.
- semilogx() ídem con escala lineal en el eje de ordenadas y logarítmica en el eje de abscisas.
- semilogy() ídem con escala lineal en el eje de abscisas y logarítmica en el eje de ordenadas.

- Existen funciones orientadas a añadir títulos al gráfico, a los ejes, a dibujar una cuadrícula auxiliar, a introducir texto, etc.
- title('título') añade un título al dibujo
- xlabel('tal') añade una etiqueta al eje de abscisas. Con xlabel off desaparece
- ylabel('cual') idem al eje de ordenadas. Con ylabel off desaparece
- text(x,y,'texto') introduce 'texto' en el lugar especificado por las coordenadas x e y. Si x e y son vectores, el texto se repite por cada par de elementos.
- grid activa una cuadrícula en el dibujo. Con grid off desaparece la cuadrícula

plot(X,Y,'opción') (opción: permite elegir color y trazo de la curva)

y	yellow	.	point	-	solid
m	magenta	o	circle	:	dotted
c	cyan	x	x-mark	-.	dashdot
r	red	+	plus	--	dashed
g	green	*	star		
b	blue	s	square		
w	white	d	diamond		
k	black	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		

Ej1

```
>> x = 1:0.1:10;

>>  y = x .* exp(-x .^2);  % Los 100 valores de y

>>   plot(x,y)
```

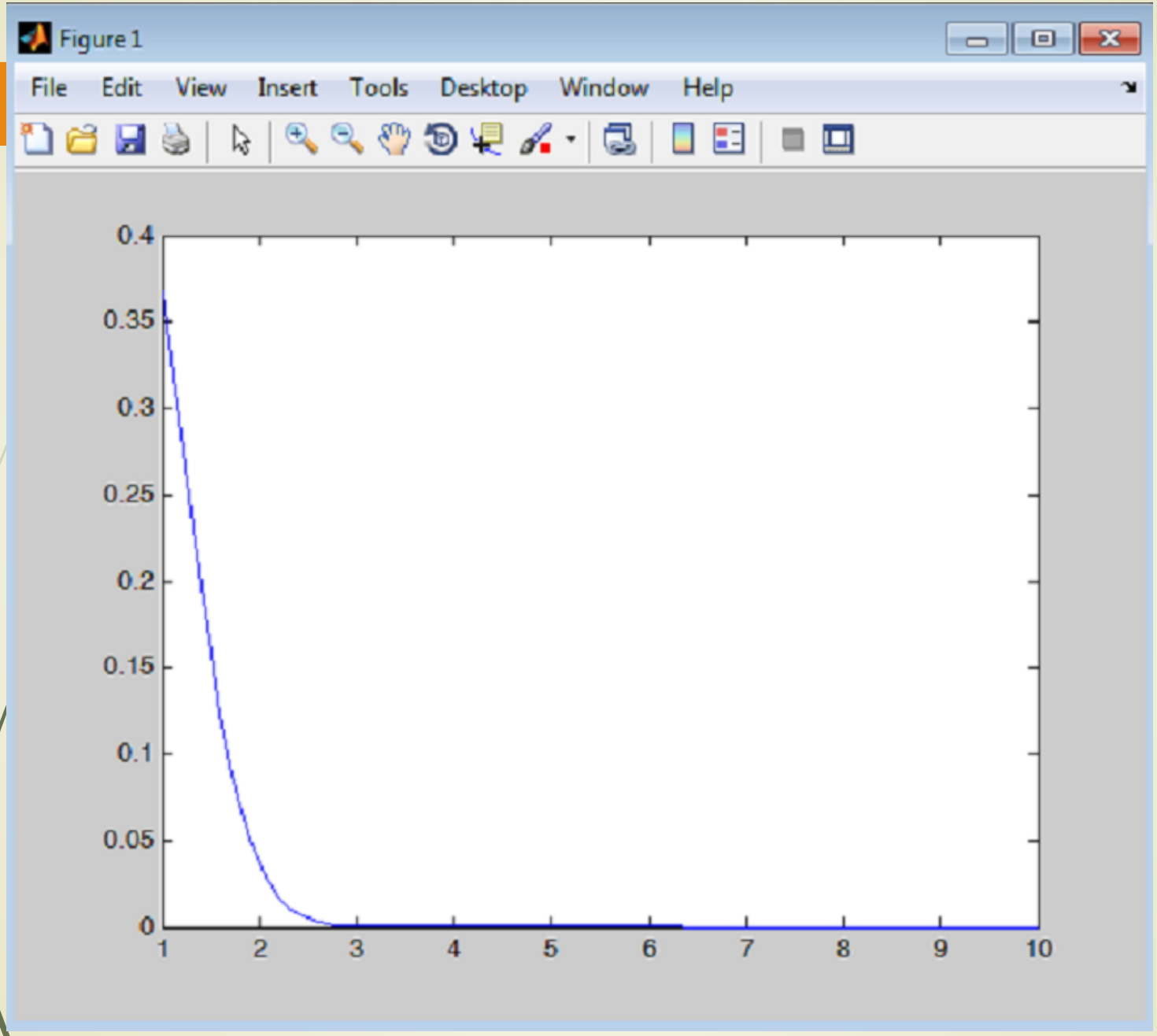


Figure 1
File Edit View Insert Tools Desktop Window Help
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1 2 3 4 5 6 7 8 9 10

Ej2

```
>>  x = 0: pi/100 : 2*pi

>>  y = sin(x) ;

>>   plot(x, y) ;    %  Así se dibuja la función y = sin(x)
```

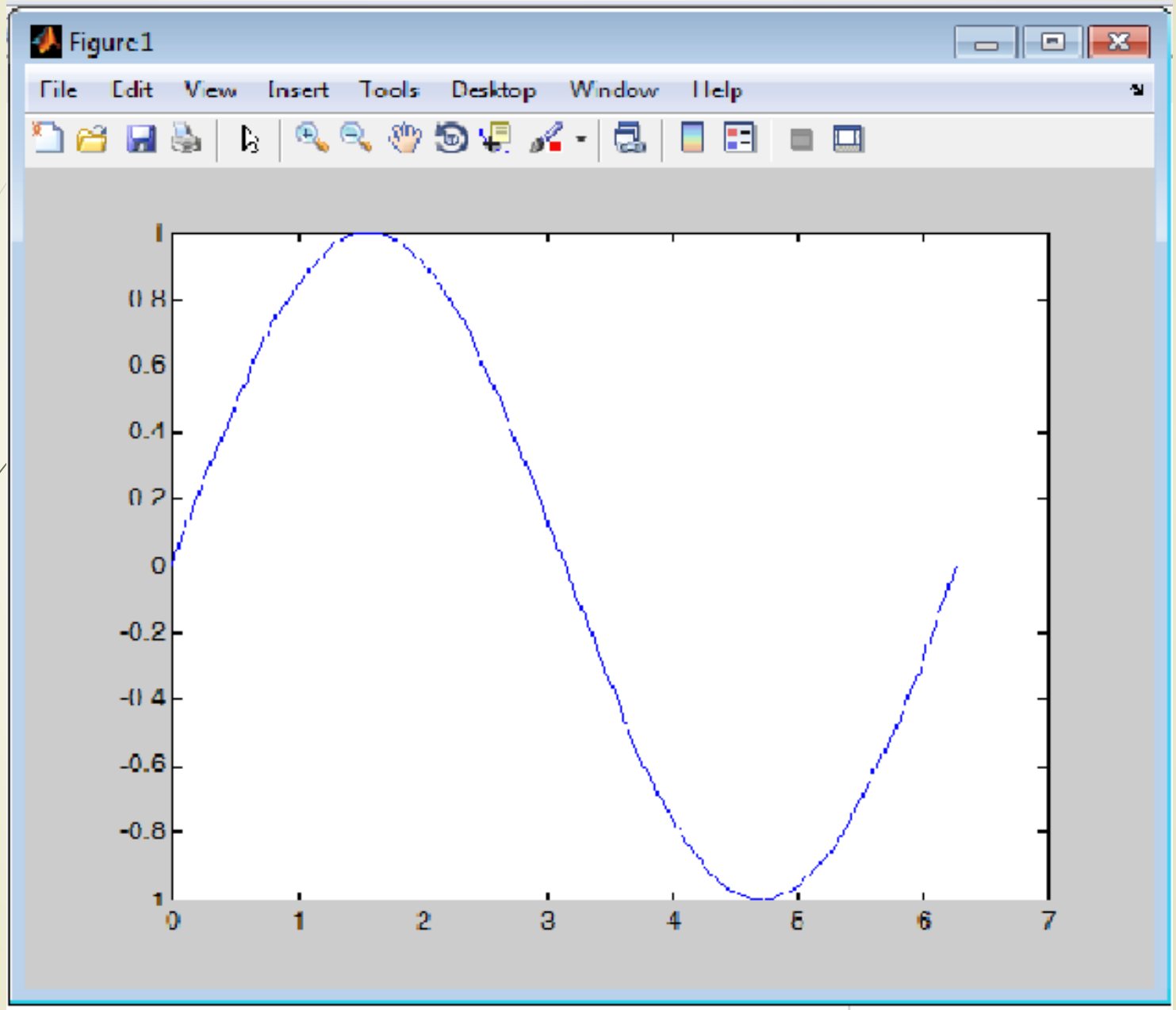


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

Ej3

```
x = linspace(0, 2*pi, 100);
y = sin(x); plot(x,y);
grid on
```

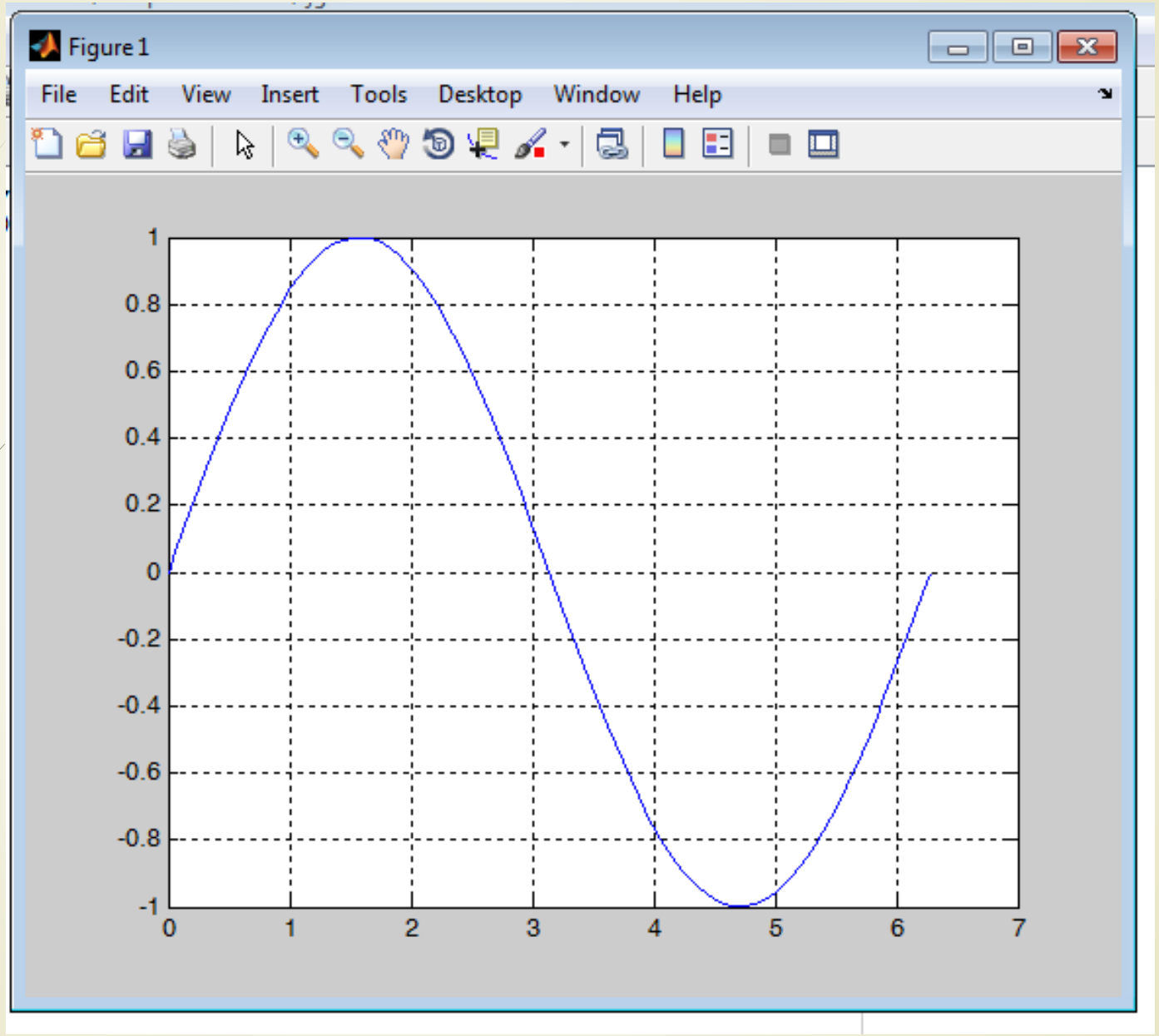


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 1 2 3 4 5 6 7

Ej4

Un ejemplo:

Sea la función:

$$f(x) = \begin{cases} x^2 & si\ x < 0 \\ 1 & si\ 0 \le x < 1 \\ -x + 2 & si\ x \ge 1 \end{cases}$$

Vamos a representarla en el intervalo (-5, +5)

>> 2 < 5

>> ans =

1 (verdadero)

>> 1 >500

>> ans =

0 (falso)

```
>> x = 1:7

    x =

        1   2   3   4   5   6   7
```

```
>> x > 4

>> ans =

        0   0   0   0   1   1   1 (donde se cumple la condición hay 1)
```

$$f(x) = \begin{cases} x^2 & si\ x < 0 \\ 1 & si\ 0 \le x < 1 \\ -x + 2 & si\ x \ge 1 \end{cases}$$

```
x = linspace(-2, 3, 3000);

y = (x.^2).*(x<0)+1.*((0<=x)&(x<1))+
(-x+2).*(x>=1);

plot(x,y,'.'), grid on
```

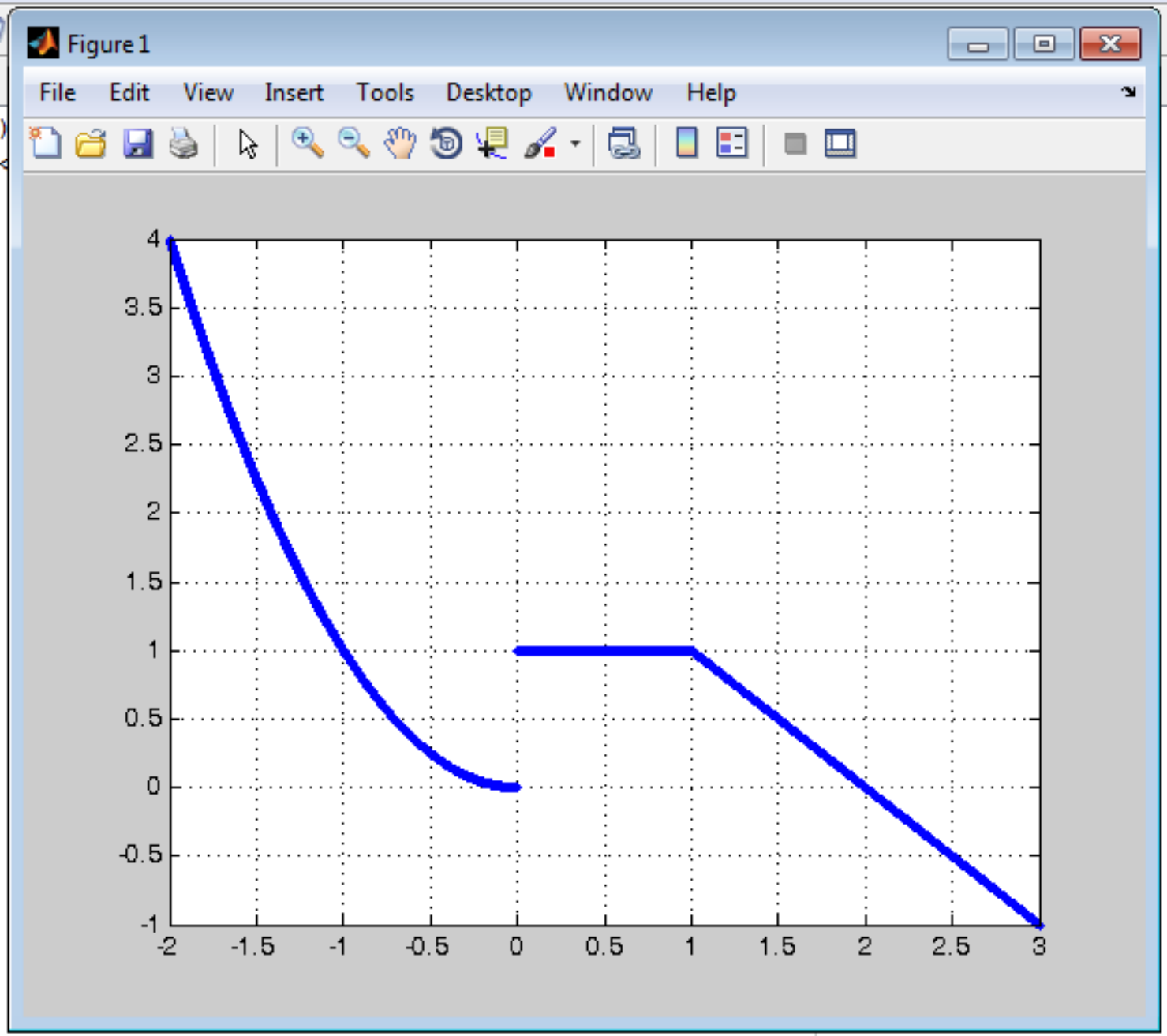
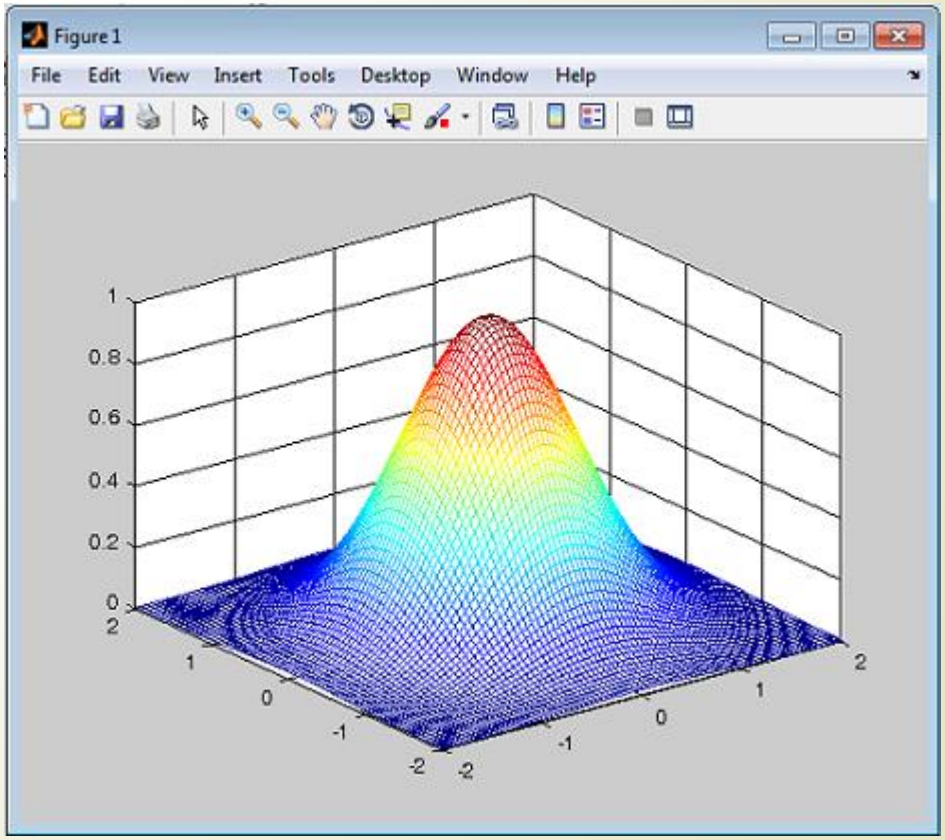


Figure 1
File Edit View Insert Tools Desktop Window Help
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 3

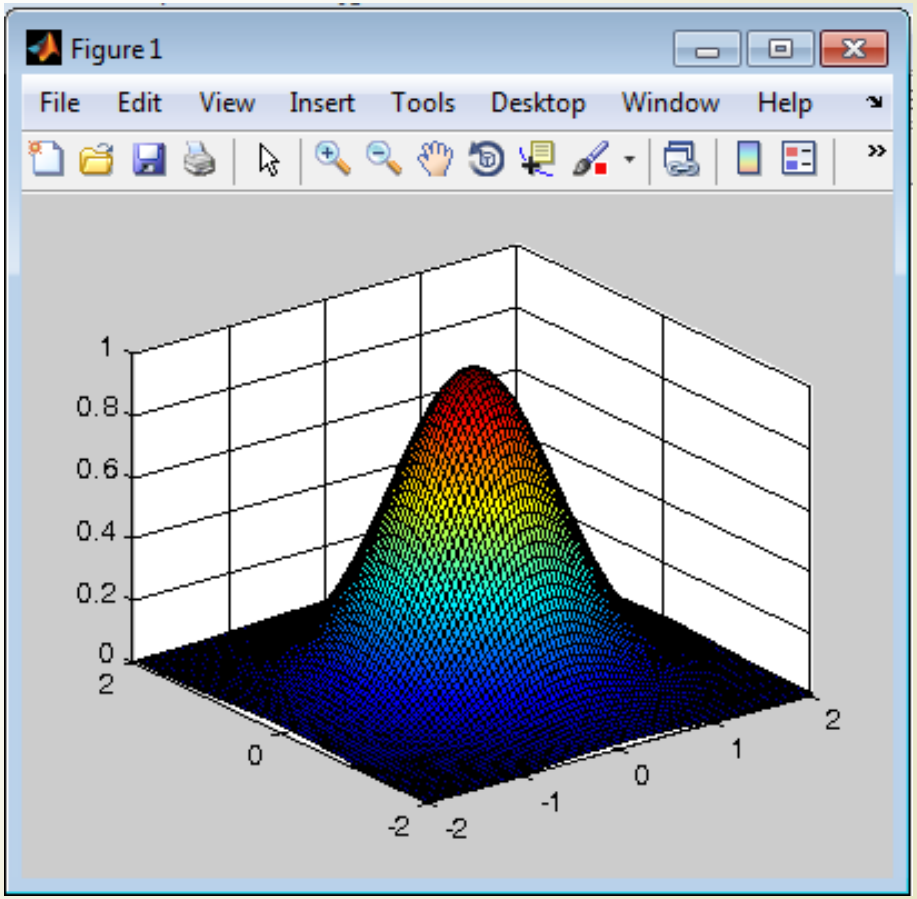
Graficas en 3D: mesh

```
[x, y] = meshgrid(-2:0.05:2);
z = exp(-x .^2 - y .^2);
mesh(x, y, z);
```



Graficas en 3D: surf

```
[x, y] = meshgrid(-2:0.05:2);
z = exp(-x .^2 - y .^2);
surf(x, y, z);
```



subplot

- Subplot(m,n,p) divide al dibujo en m x n y dibuja en la posición indicada por p.
- El primer subplot es la primera columna de la primera fila, el segundo está en la segunda columna de la primera fila y así siguiendo.

Ejemplo subplot

```
x= -2 :0.2: 2;
y= -2 :0.2: 2;
[X,Y]=meshgrid(x,y);
Z= X.*exp(-X.^2 - Y.^2);
subplot(1,2,1);
mesh(X,Y,Z),title('Gráfica mesh');
xlabel('eje x'), ylabel('eje y'), zlabel('eje z');
subplot(1,2,2);
surf(X,Y,Z), title('Gráfica de superficie');
xlabel('eje x'), ylabel('eje y'),zlabel('eje z');
```

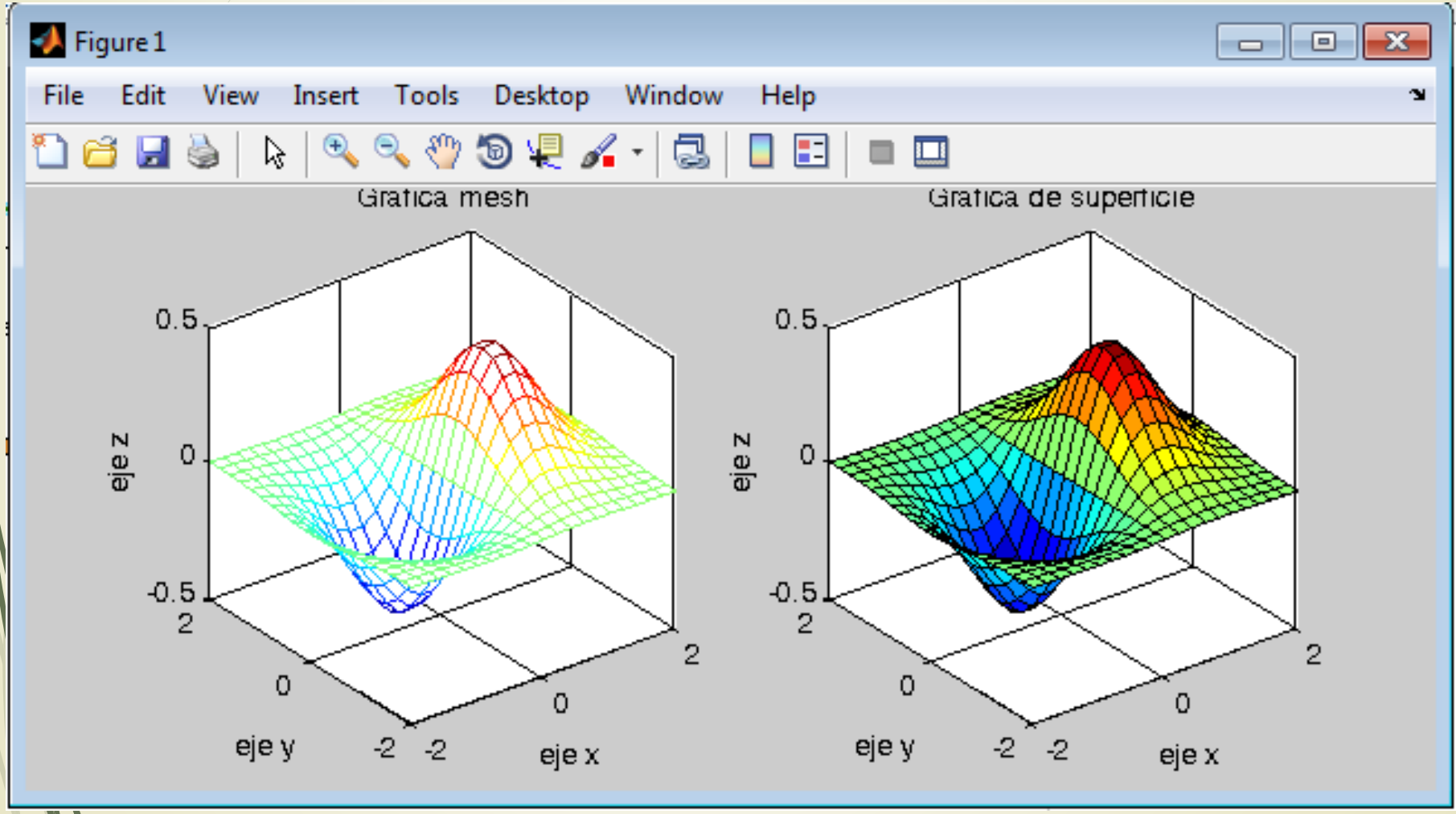


Figure 1
File Edit View Insert Tools Desktop Window Help
Gratica mesh
Gratica de superficie
0.5
0
-0.5
eje z
2
0
-2
eje y
-2
0
2
eje x

Estructuras de control: condicionales

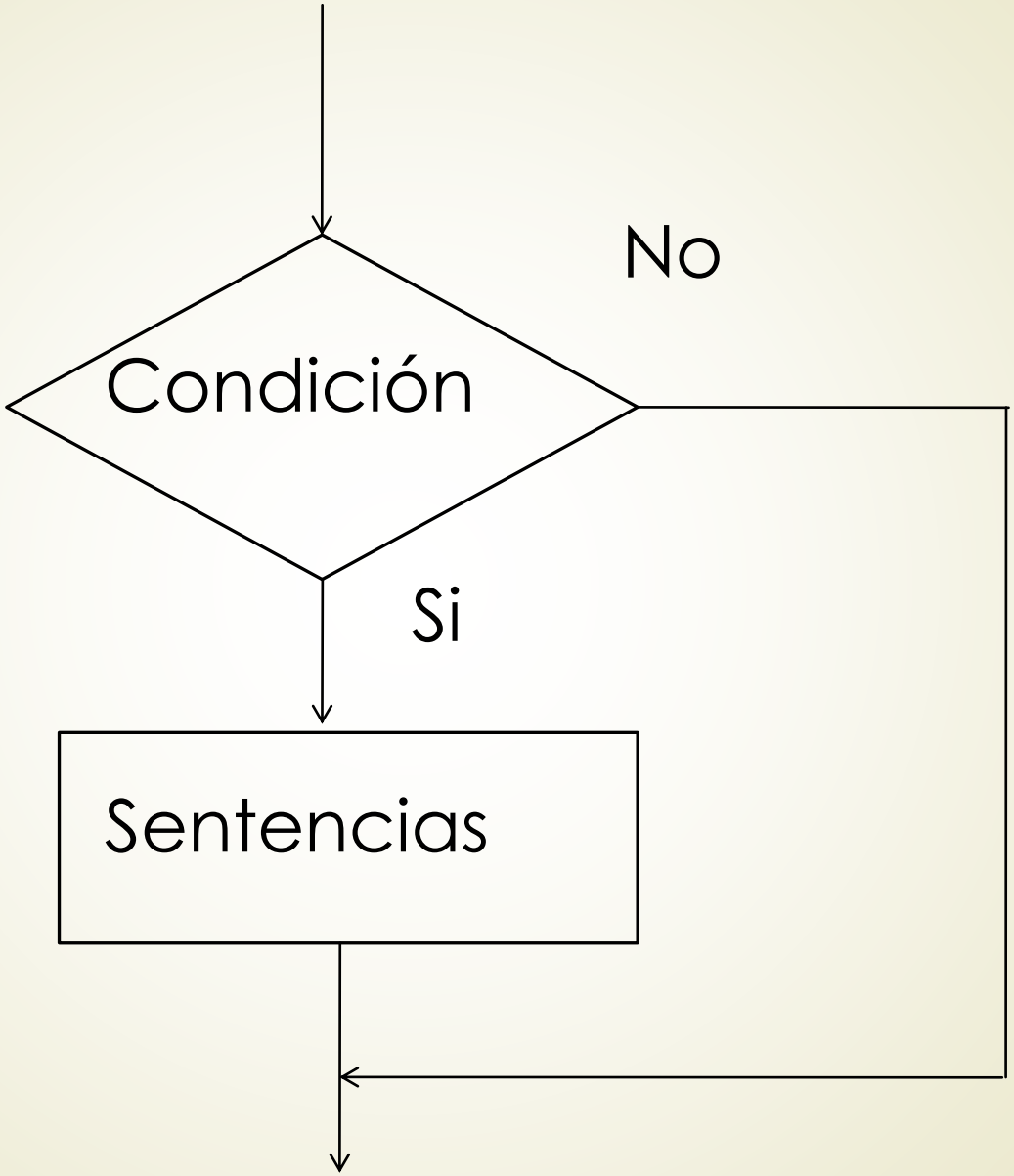
- Una sentencia condicional es una instrucción MATLAB que permite tomar decisiones sobre si se ejecuta un grupo de comandos que cumplen una condición o, por el contrario omitirlos.
- En una sentencia condicional se evalúa una expresión condicional. Si la expresión es verdadera, el grupo o bloque de comandos se ejecutan. Si la expresión es falsa, MATLAB no ejecuta (salta) el grupo de comandos en cuestión.
- Las sentencias condicionales pueden ser parte de un sctipt de una función

if end

- Una estructura if simple tiene la siguiente forma:

```
if  (condición)
 sentencias
end
```

- Si la condición (una expresión lógica) es verdadera, se ejecutan las sentencias y sigue a end.
- Si la comparación es falsa, el programa salta inmediatamente a la sentencia que sigue a end.


No
Condición
Si
Sentencias

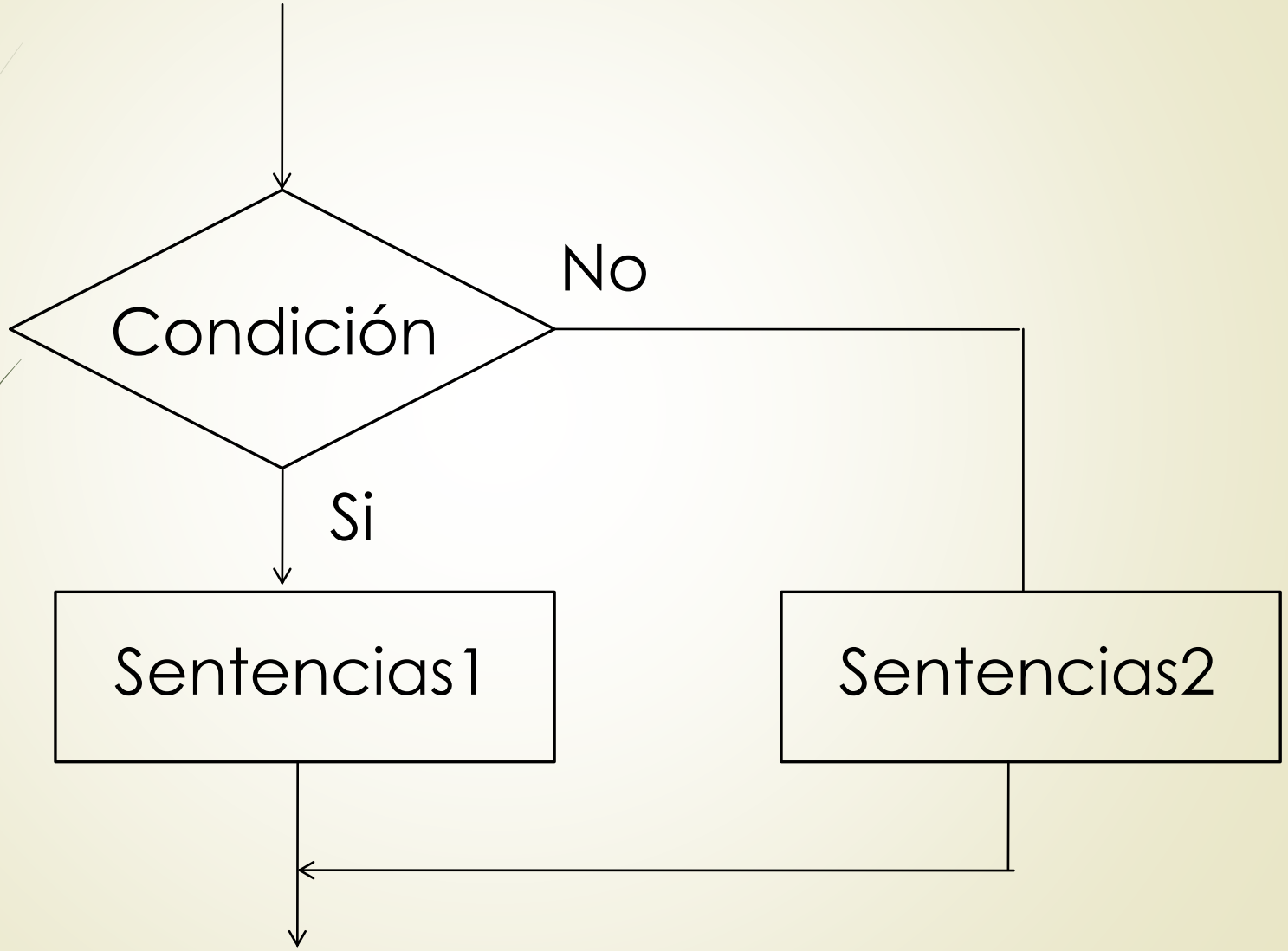
if else end

- El if simple le permite ejecutar una serie de sentencias si una condición es verdadera y saltar dichos pasos si la condición es falsa.
- La cláusula else le permite ejecutar un conjunto de sentencias1 si la comparación es verdadera y un conjunto diferente de sentencias2 si la comparación es falsa.

```
if  (condición)
      sentencias1
else
      sentencias2
end
```

else

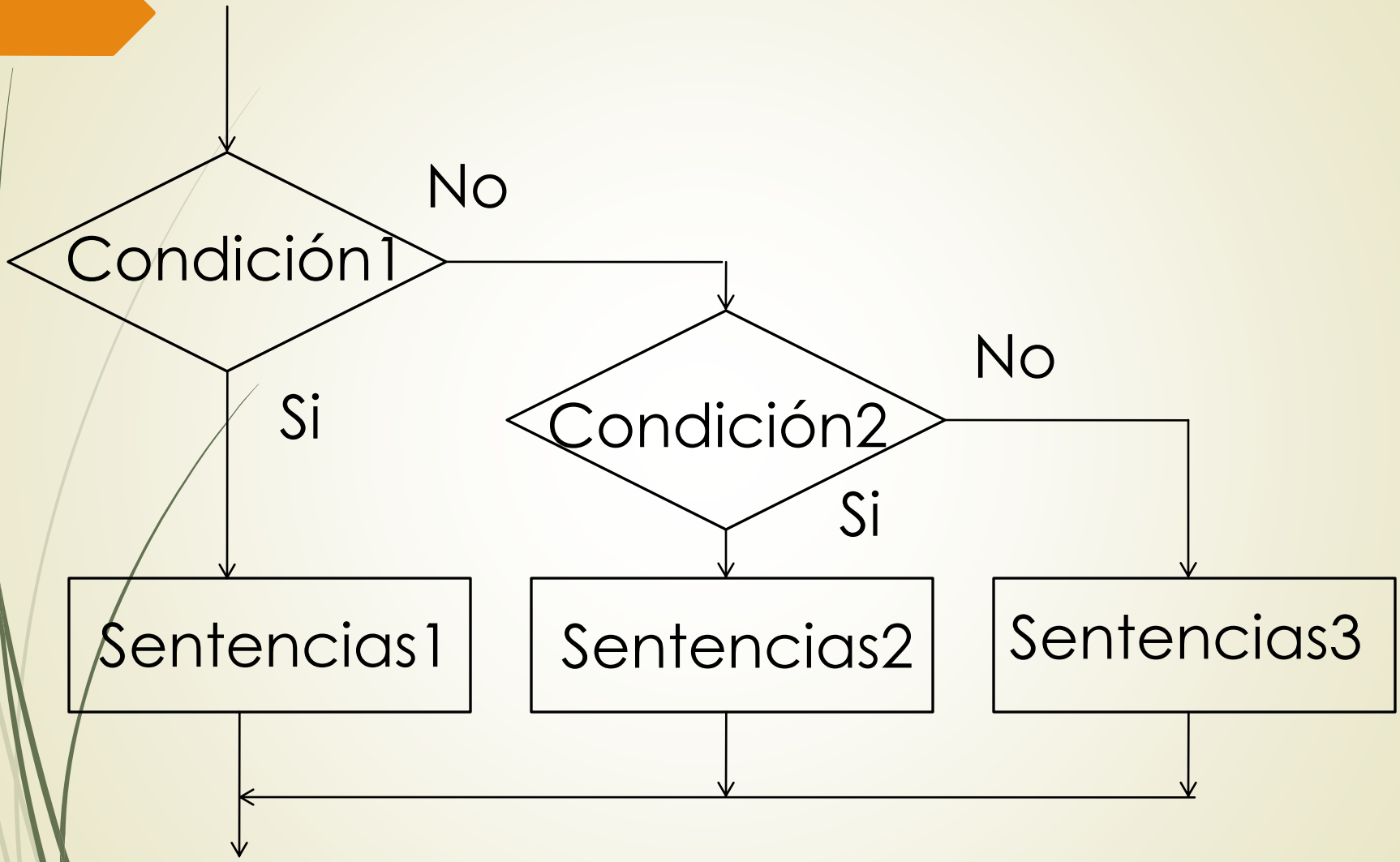
- ✓ Recordar que la sentencia else es opcional. Esto significa que en el caso de que haya varios elseif y ningún else si alguna condición de los elseif es verdadera, los comandos serán ejecutados, pero en otro caso (todas las condiciones de los elseif son falsas) no se ejecutan ni se realizará ninguna operación.


Condición
No
Si
Sentencias1
Sentencias2

if elseif else end

- ✓ Esta estructura incluye dos condicionales, lo que hace posible ejecutar uno de entre tres grupos de instrucciones diferentes.

```
if  (condicion1)
      sentencias1
elseif  (condicion2)
      sentencias2
else
      sentencias3
end
```


Condición1
No
Si
Condición2
No
Si
Sentencias1
Sentencias2
Sentencias3

Switch y case

- La estructura switch-case se usa con frecuencia cuando existe una serie de opciones de ruta de programación para una variable dada, dependiendo de su valor.
- Switch-case es similar a if/else/elseif. De hecho, cualquier cosa que pueda hacer con switch/case se podría hacer con if/else/elseif.
- Se trata de una estructura que permite elegir entre múltiples salidas, con base en ciertos criterios. Los criterios pueden ser un escalar (un número) o una cadena.

Switch variable

case opcion1

código a ejecutar si la variable es igual a opcion1

case opcion2

código a ejecutar si la variable es igual a opcion2

.....

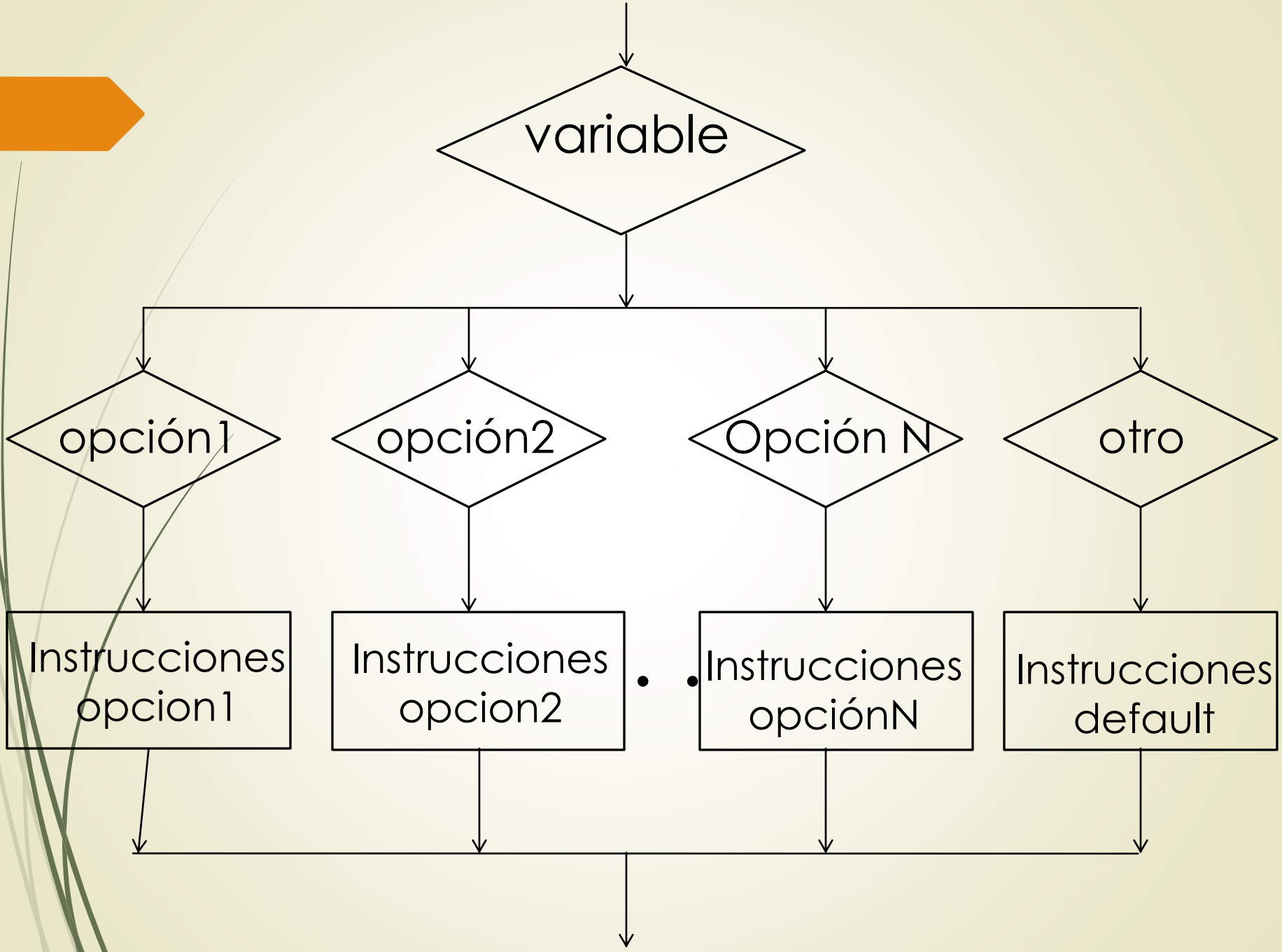
case opcionN

código a ejecutar si la variable es igual a opcionN

otherwise

código a ejecutar por default

end


variable
opción1
opción2
Opción N
otro
Instrucciones opcion1
Instrucciones opcion2
Instrucciones opciónN
Instrucciones default

Estructuras de repetición: lazos (bucles)

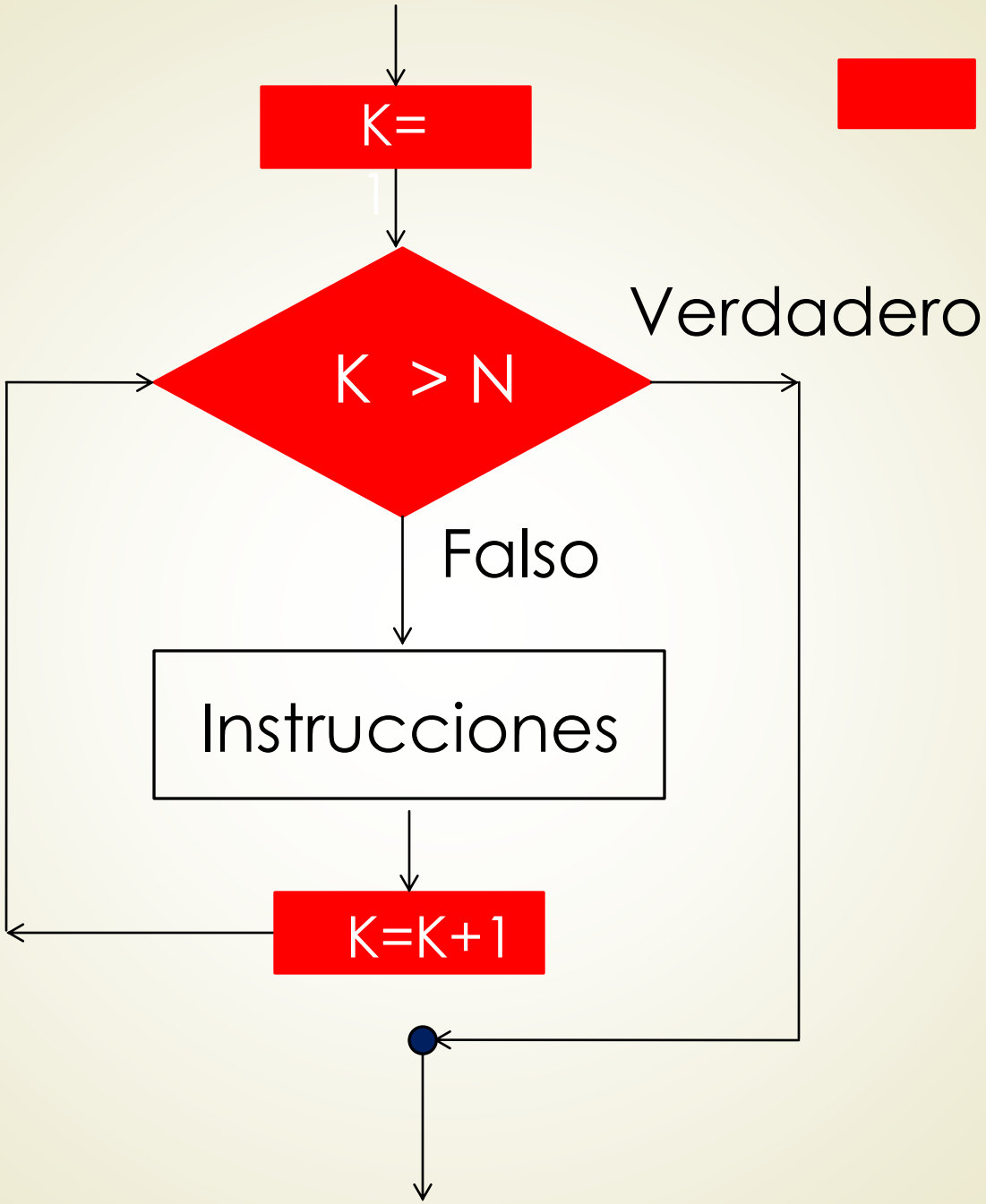
- ✓ Los bucles permiten repetir las mismas operaciones sobre datos distintos.
- ✓ Mientras que en C/C++ el "cuerpo" de estas sentencias se determinaba mediante llaves {...}, en MATLAB se utiliza la palabra end con la misma finalidad.
- ✓ MATLAB soporta dos tipos diferentes de bucles: el bucle for y el bucle while.

For

- Estructura: la primera línea identifica el bucle y define un índice, que es un número que cambia en cada paso a través del bucle. Después de la línea de identificación viene el grupo de comandos que se quiere ejecutar. Finalmente, la terminación del bucle se identifica mediante el comando end.

```
for  índice = inicial : incremento : final
        comandos
end
```

- El bucle se ejecuta una vez para cada elemento del índice identificado en la primera línea.



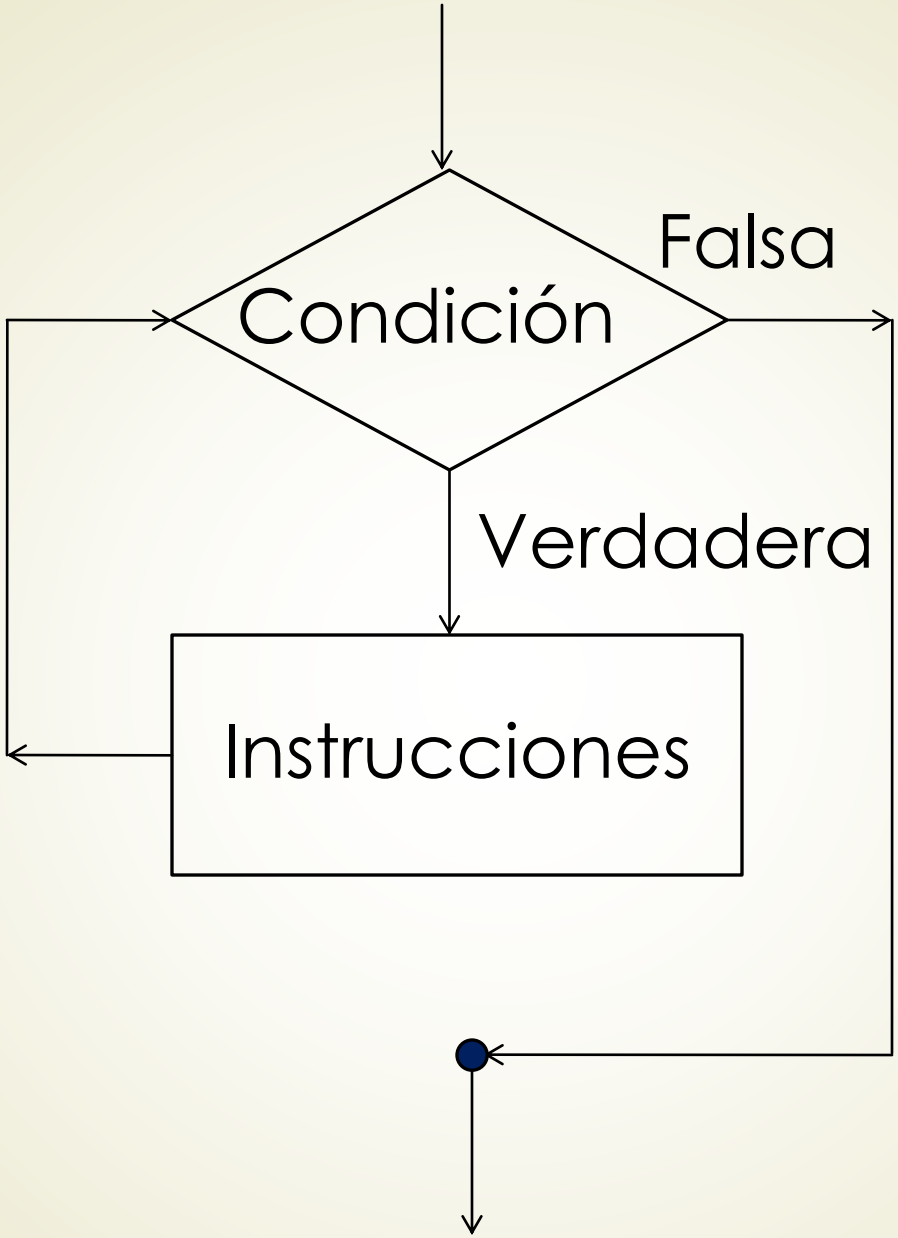
= for

while

- El bucle se repite mientras se cumpla la condición (mientras sea verdadera).

```
While (condición)
    comandos
end
```

- El for se ejecuta un número determinado de veces, el while no sabemos. Depende que se cumpla la condición.


Falsa
Condición
Verdadera
Instrucciones