Introducción a MATLAB

Departamento de Ingeniería de Sistemas y Automática
EUITI de Bilbao

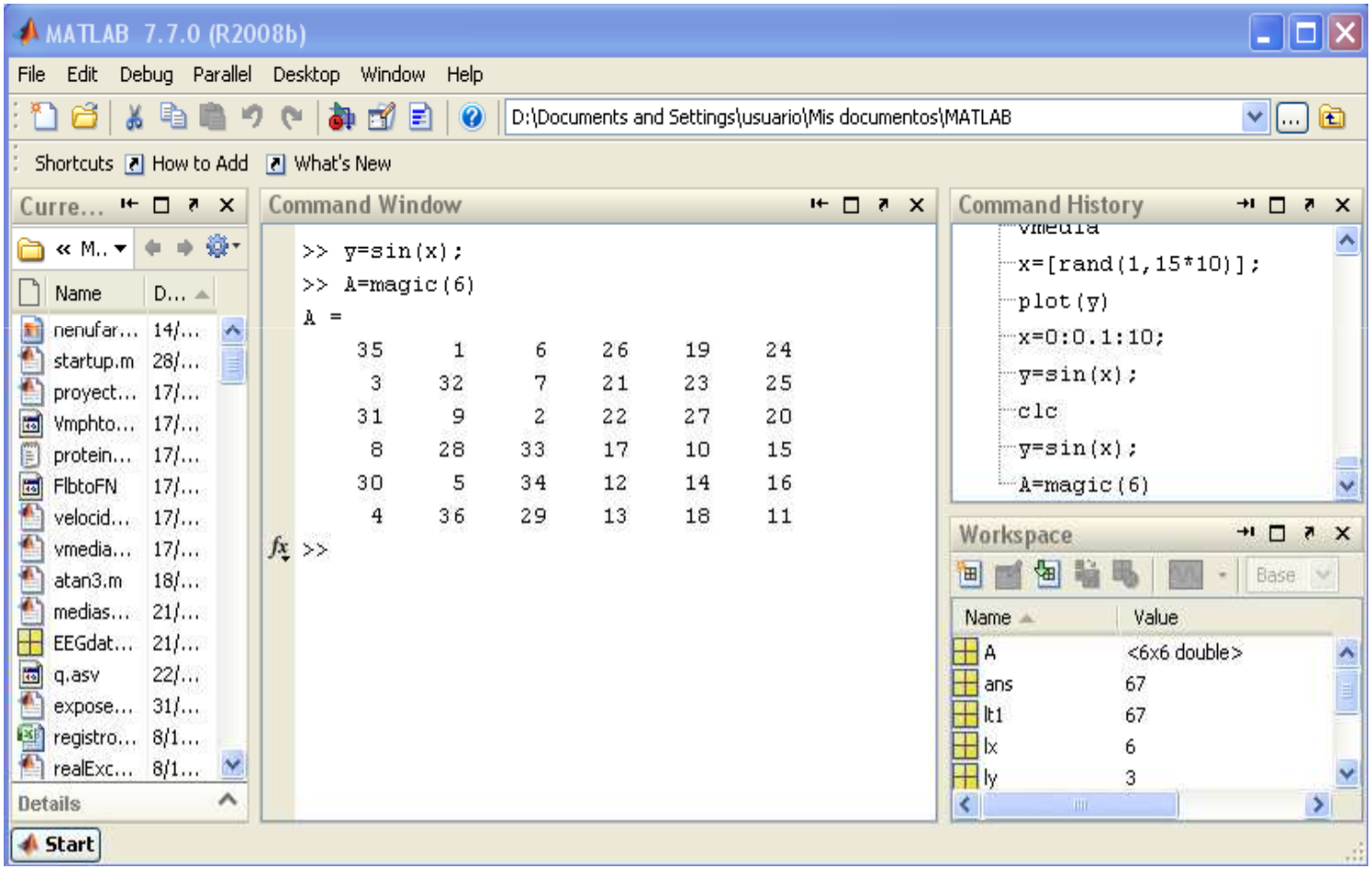
Índice

Índice

Introducción

- MATLAB es el nombre abreviado de “MATrix LABoratory”.
- Programa que realiza :
  - Cálculos numéricos con vectores y matrices.
  - Gráficos en 2 y 3 dimensiones.
- Dispone de:
  - Código básico
  - Librerías especializadas (toolboxes)
- Arrancar:
  - Por medio del menú Inicio
  - Clicando dos veces en el icono

Introducción



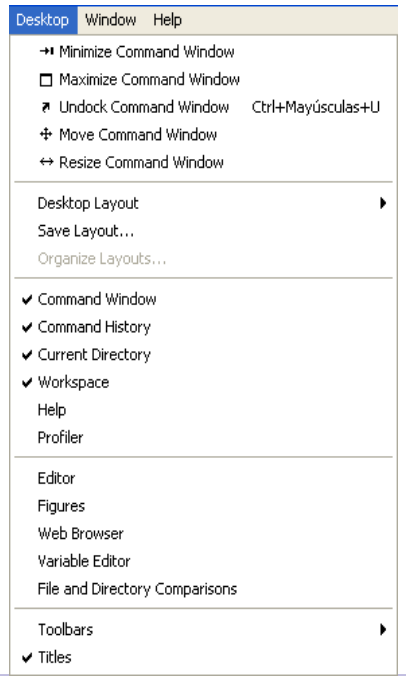
Índice

Entorno de Trabajo

1. Escritorio de Matlab (Matlab Desktop): ventana o contenedor de máximo nivel en la que se pueden situar (to dock) los demás componentes.
2. Los componentes individuales, orientadas a tareas concretas, entre las que se puede citar:
   - La ventana de comandos (*Command Window*),
   - La ventana histórica de comandos (*Command History*),
   - El espacio de trabajo (*Workspace*),
   - La plataforma de lanzamiento (*Launch Path*),
   - El directorio actual (*Current Directory*),
   - La ventana de ayuda (*Help*)
   - El editor de ficheros y depurador de errores (*Editor&Debugger*),
   - El editor de vectores y matrices (*Array Editor*).
   - La ventana que permite estudiar cómo se emplea el tiempo de ejecución (*Profiler*).

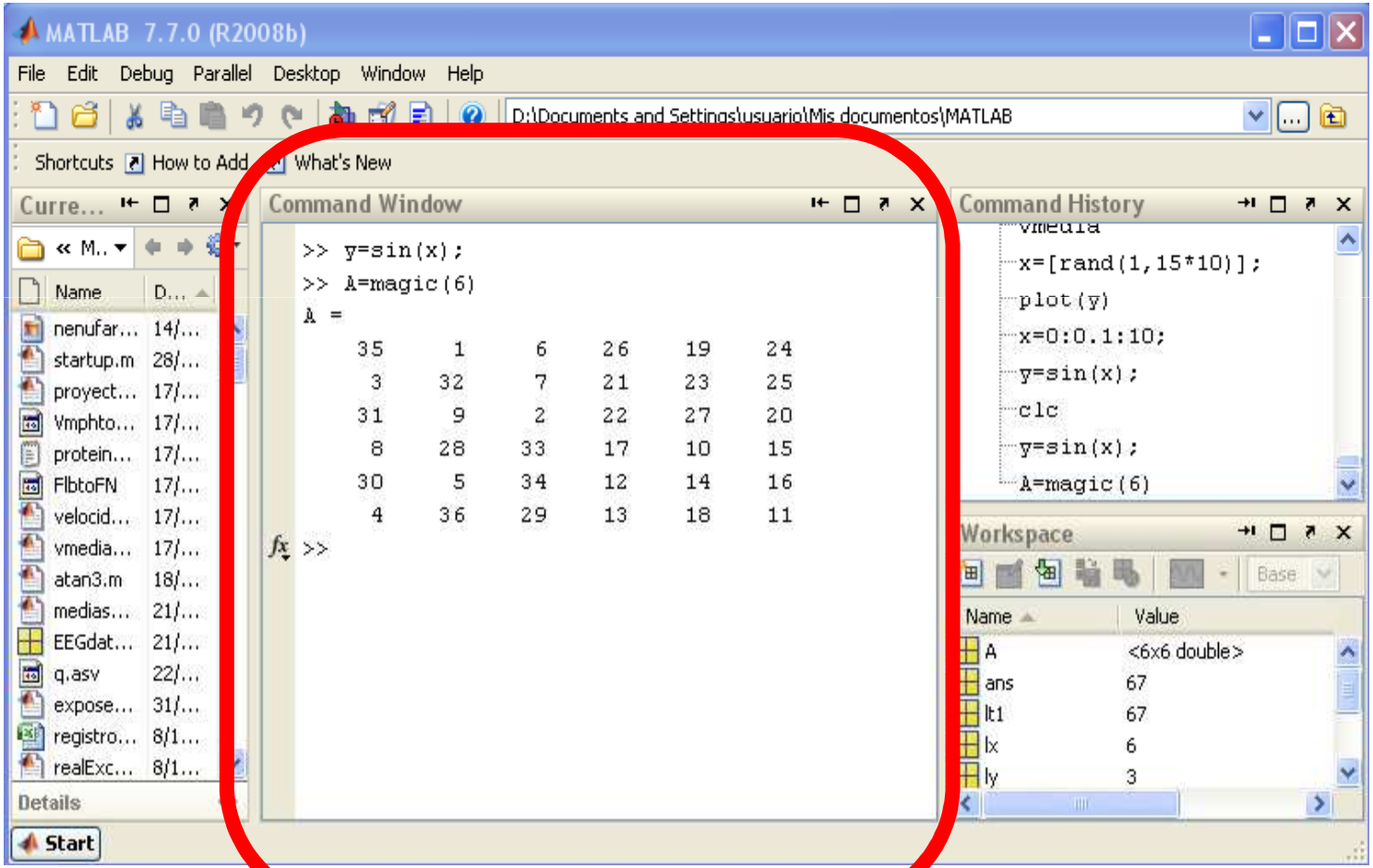
Entorno de Trabajo

- *ESCRITORIO Matlab*
  - La ventana más general de la aplicación.
  - Donde se organizan el resto de ventanas.
  - Para elegir la configuración desde el menú Desktop



Entorno de Trabajo

- *COMMAND WINDOW*



Entorno de Trabajo

- *COMMAND WINDOW*
  - Ejecuta interactivamente las instrucciones de MATLAB y muestra los resultados correspondientes.
  - Para teclear un comando el cursor debe estar situado después del símbolo '>>' *prompt*. Para ejecutarlo Intro.

```
>> 9
ans =
    9
```

```
>> 5+3
ans =
    8
```

  - No es posible ir hacia arriba, a una línea anterior, o realizar una corrección y reejecutar un nuevo comando.
  - Es posible recuperar comandos anteriores y moverse por dichos comandos con el ratón y con las teclas-flechas ↑ y ↓.
  - Para moverse sobre la línea de comandos : ← y →

Entorno de Trabajo

- *COMMAND WINDOW*
  - El punto y coma (;)

    Si se teclea un (;) al final de un comando, la salida de dicho comando no será visualizada.

```
>> x=9
x =
    9
>> x=9;
>>
```

  - La variable ans

    Cuando el resultado de la operación no ha sido asignado a ninguna variable, MATLAB utiliza un nombre de variable por defecto (*ans*, de *answer*).

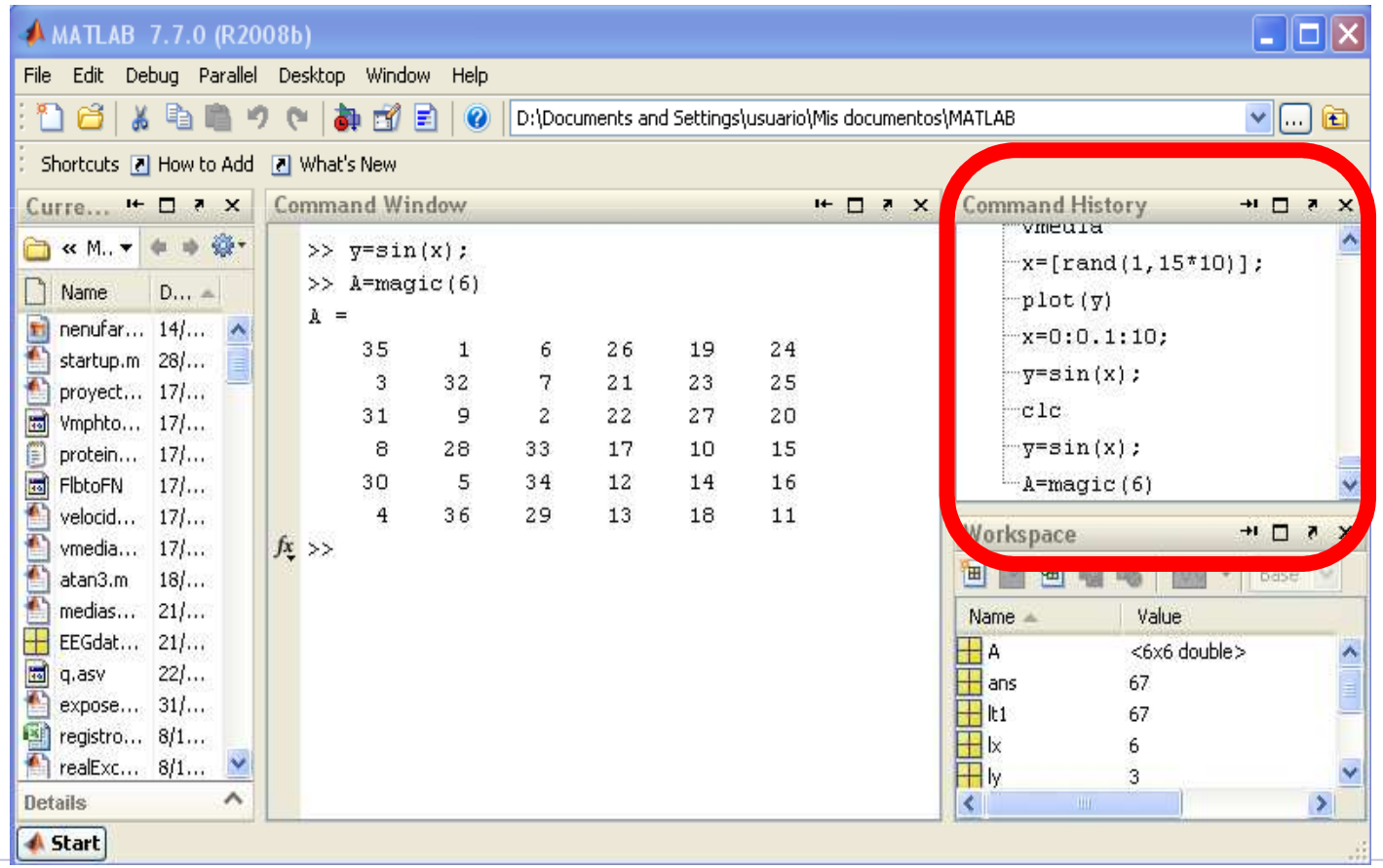
```
>> 9
ans =
    9
>> ans+1
ans =
   10
```

Entorno de Trabajo

- *COMMAND WINDOW*
  - El símbolo (%)
    - Cuando se teclea el símbolo % al principio de una línea, Matlab considerará dicha línea como un comentario y no la ejecutará.
    - No se utiliza en Command Window, pero si en los programas.
  - Comando clc
    - Clc (clear console) limpia la pantalla de Command Window pero mantiene las variables usadas.

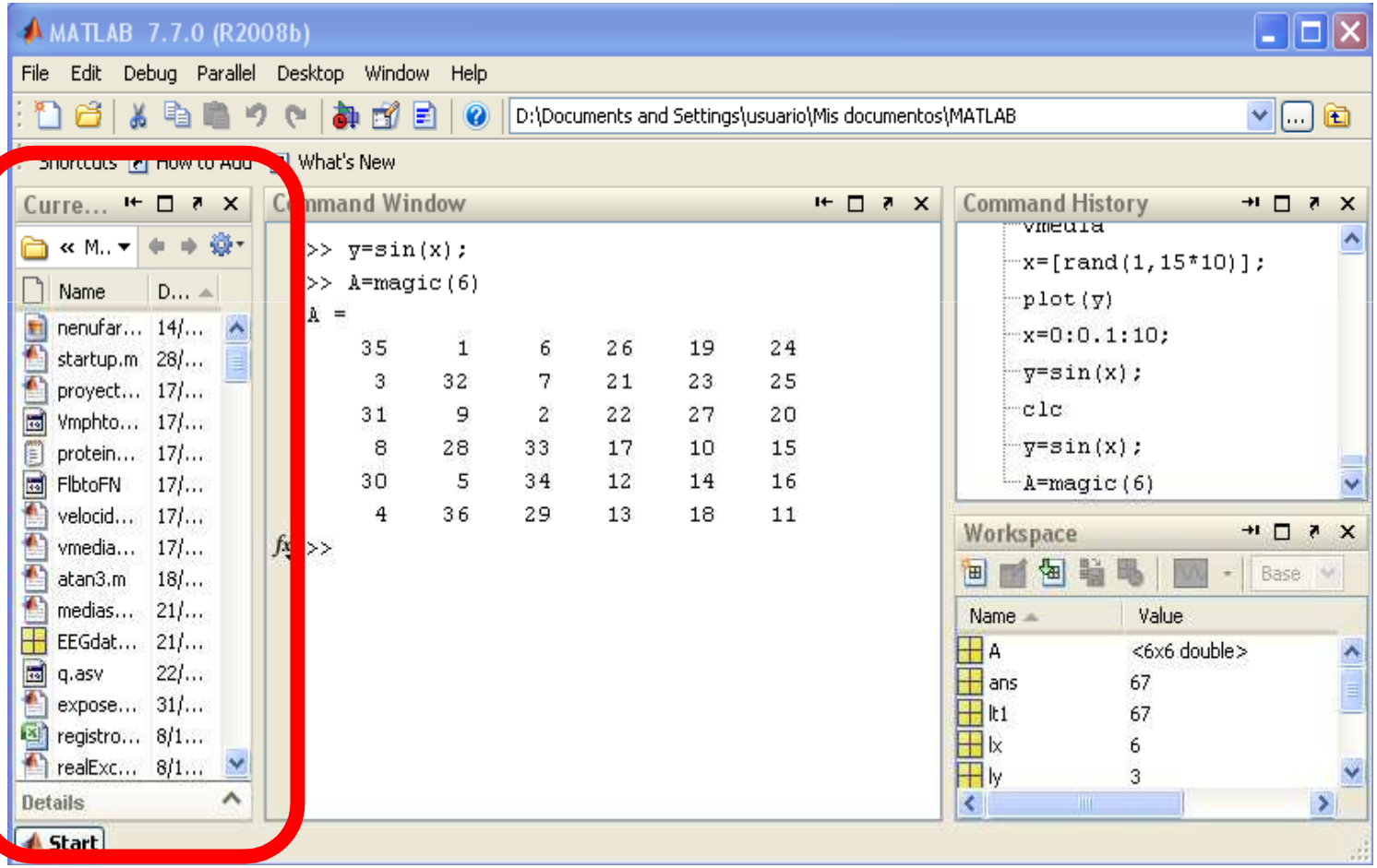
Entorno de Trabajo

- *COMMAND HISTORY BROWSER*
  - *Acceso a sentencias ejecutadas en Command Window.*
  - *Para volver a ejecutarlas, doble clic en la sentencia.*



Entorno de Trabajo

- *CURRENT DIRECTORY BROWSER*



Entorno de Trabajo

- *CURRENT DIRECTORY BROWSER*
  - *Este directorio es el primer sitio en el que MATLAB busca cuando desde la línea de comandos se le pide que ejecute un fichero.*
  - *El Comando pwd: permite saber cual es el directorio actual.*

```
>> pwd
ans =
   D:\Documents and Settings\usuario\Mis documentos\MATLAB
```

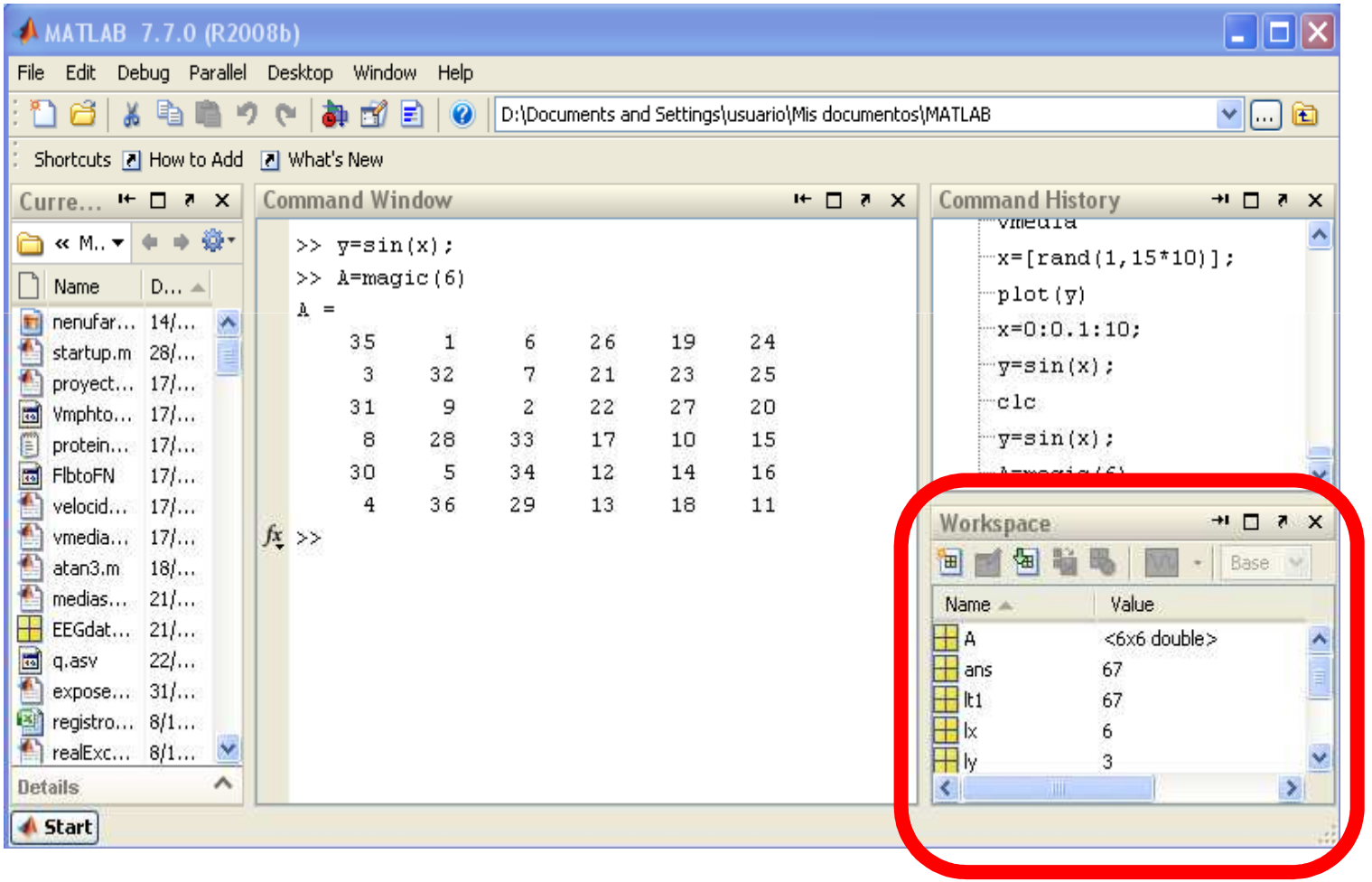
  - *El comando cd: Cambia el directorio*

```
>> cd C:\Matlab\Ejemplos
```

  - *Desde este directorio se pueden explorar las carpetas como desde cualquier aplicación de Window.*
  - *Abrir un programa: doble clic*

Entorno de Trabajo

- *WORKSPACE BROWSER*



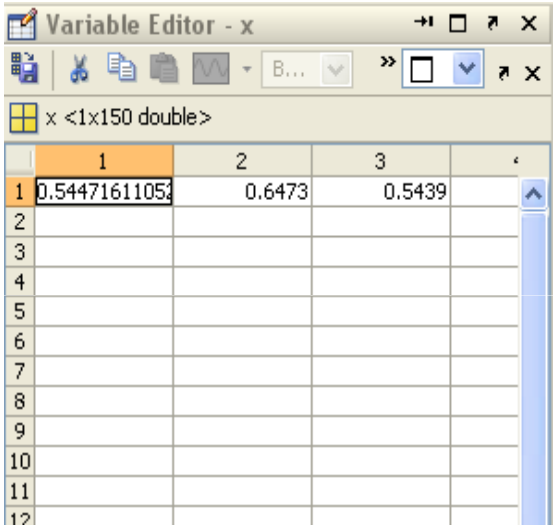
Entorno de Trabajo

- *WORKSPACE BROWSER*
  - *Conjunto de variables y funciones definidas en la memoria del programa o de la función que se está ejecutando (espacio de trabajo).*
  - *Cada función tiene su propio espacio de trabajo.*
  - *Cuando se termina de ejecutar la función, desaparecen las variables de dicha función a no ser que se hayan declarado como persistentes.*
  - *Comando whos: información de las variables por línea de comandos.*

```
>> whos
Name Size Bytes Class
A 3x3 72 double array
B 3x3 72 double array
C 3x3 72 double array
D 3x3 72 double array
Grand total is 36 elements using 288 bytes
```

Entorno de Trabajo

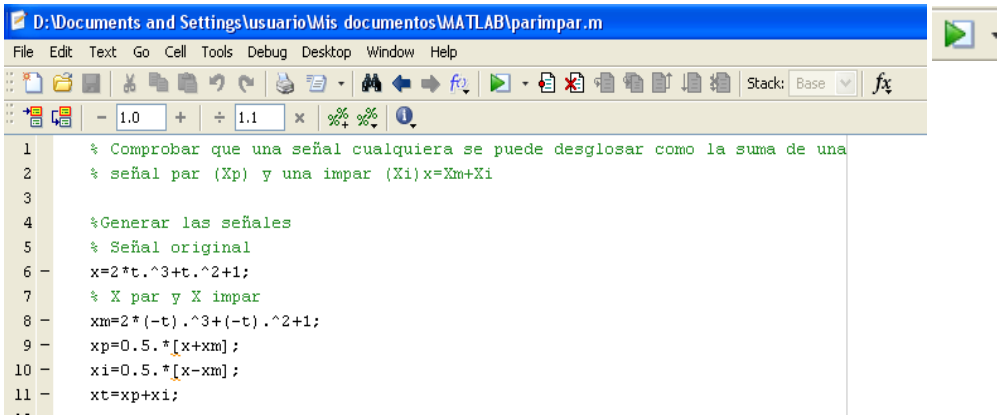
- *ARRAY EDITOR*
  - *Aparece haciendo doble clic sobre cualquier variable en el Workspace.*



  - *Se puede ver y modificar los valores de la variable.*
  - *Útil para entender algoritmos (como varían el valor de las variables) usando Debugger*

Entorno de Trabajo

- *EDITOR /DEBUGGER*
  - *Ficheros *.m (ficheros-M)*
    - *De texto ASCII.*
    - *Contienen conjuntos de comandos o definición de funciones.*
    - *Al teclear su nombre desde la línea de comandos y Enter -> se ejecutan uno a uno el contenido del fichero.*
  - *Editor*
    - *Crea , modifica y ejecuta los ficheros-M.*
    - *Para ejecutar el fichero-M* -> F5, o botón Run



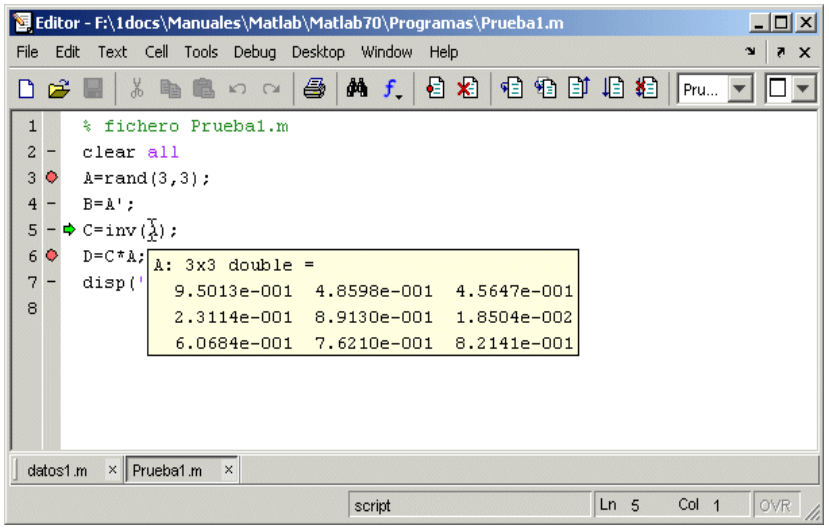
Entorno de Trabajo

- *EDITOR /DEBUGGER*

  Debugger: (Ejecución controlada)

  Arrancar el programa y este se ejecuta hasta que encuentra un breakpoint ● donde se para.

  ➡ *Indica la parte del programa donde se está*



Botones

Set/Clear Breakpoint: Coloca o borra un *breakpoint* en la línea en que está el cursor.
Clear All Breakpoints: Elimina todos los *breakpoints* que haya en el fichero.
Step: Avanzar un paso sin entrar en las funciones de usuario llamadas en esa línea.
Step In: Avanzar un paso, y si en ese paso hay una llamada a una función cuyo fichero *.m está accesible, entra en dicha función.
Step Out: Salir de la función que se está ejecutando en ese momento.
Continue: Continuar la ejecución hasta el siguiente *breakpoint*.
Quit Debugging: Terminar la ejecución del *Debugger*.

Índice

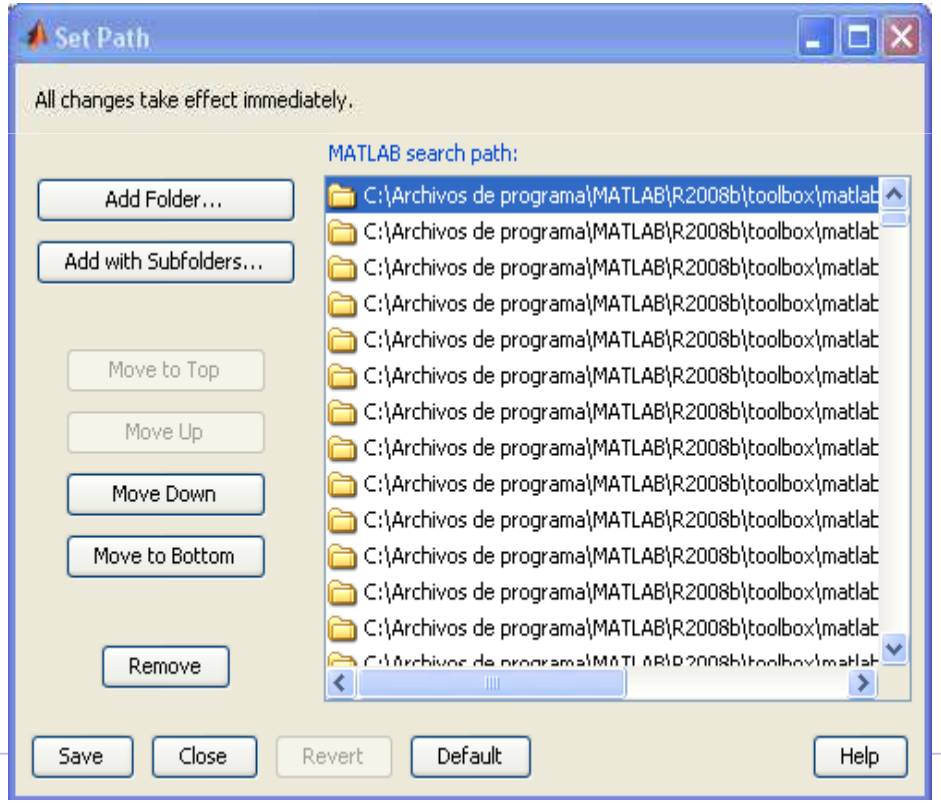
1. Introducción
2. Entorno de Trabajo
3. **Path**
4. Ayuda-Help
5. Preferencias
6. Formatos de salida
7. Comandos

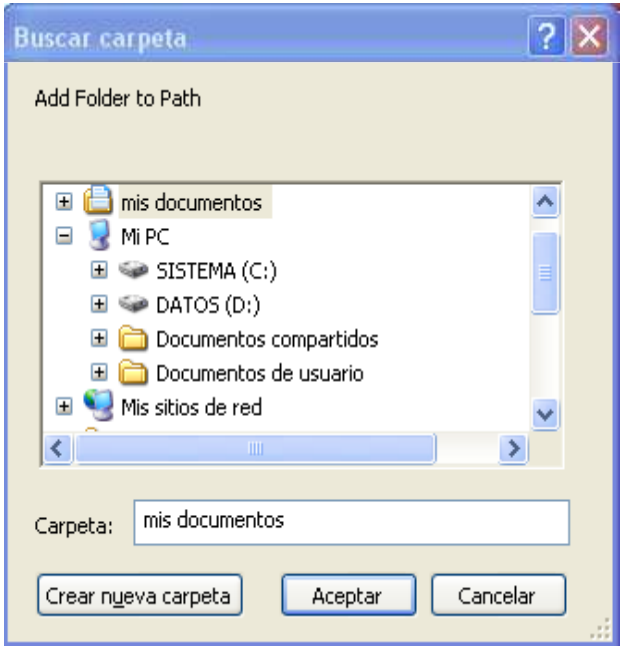
Path

- MATLAB puede llamar a una gran variedad de funciones.
- Las reglas que determinan qué función o qué fichero *.m (nombre1.m) es el que se va a ejecutar:
  1. Comprueba si *nombre1* es una variable previamente definida por el usuario.
  2. Comprueba si *nombre1* es una función interna o intrínseca de MATLAB.
  3. Comprueba si *nombre1* es una *sub-función* o una función *privada* del usuario.
  4. Comprueba si hay un fichero llamado *nombre1.mex*, *nombre1.dll* o *nombre1.m* en el *directorio actual*.
  5. Comprueba si hay ficheros llamados *nombre1.mex*, *nombre1.dll* o *nombre1.m* en los directorios incluidos en el *search path* de MATLAB.

Path

- *Search Path* (Camino de búsqueda). Lista de directorios que se puede modificar mediante:
  - Linea de comandos
  - Mediante el cuadro Set Pat en el menu *File/Set path*

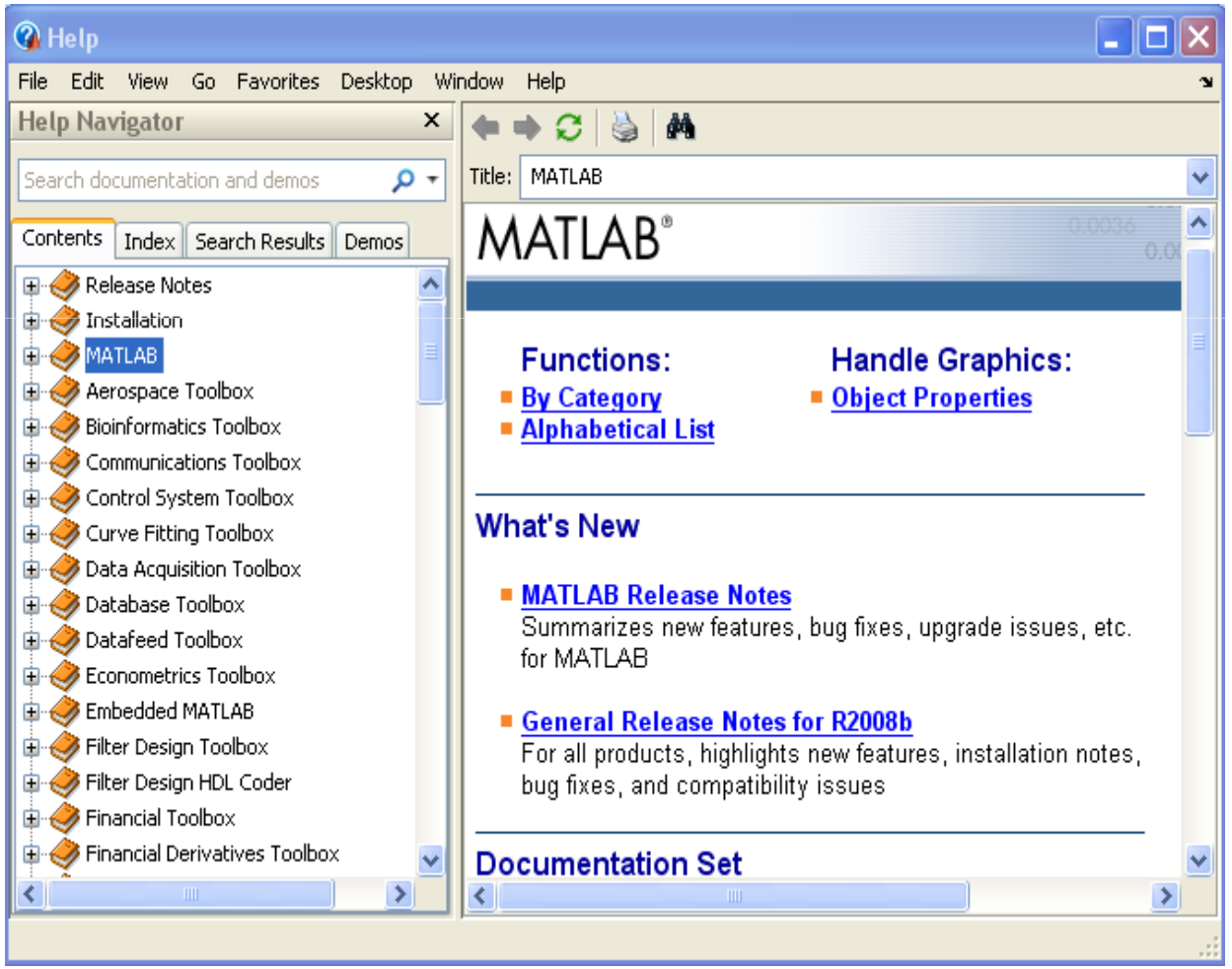




Índice

Ayuda-Help

- Menu/Help/Product Help



Ayuda-Help

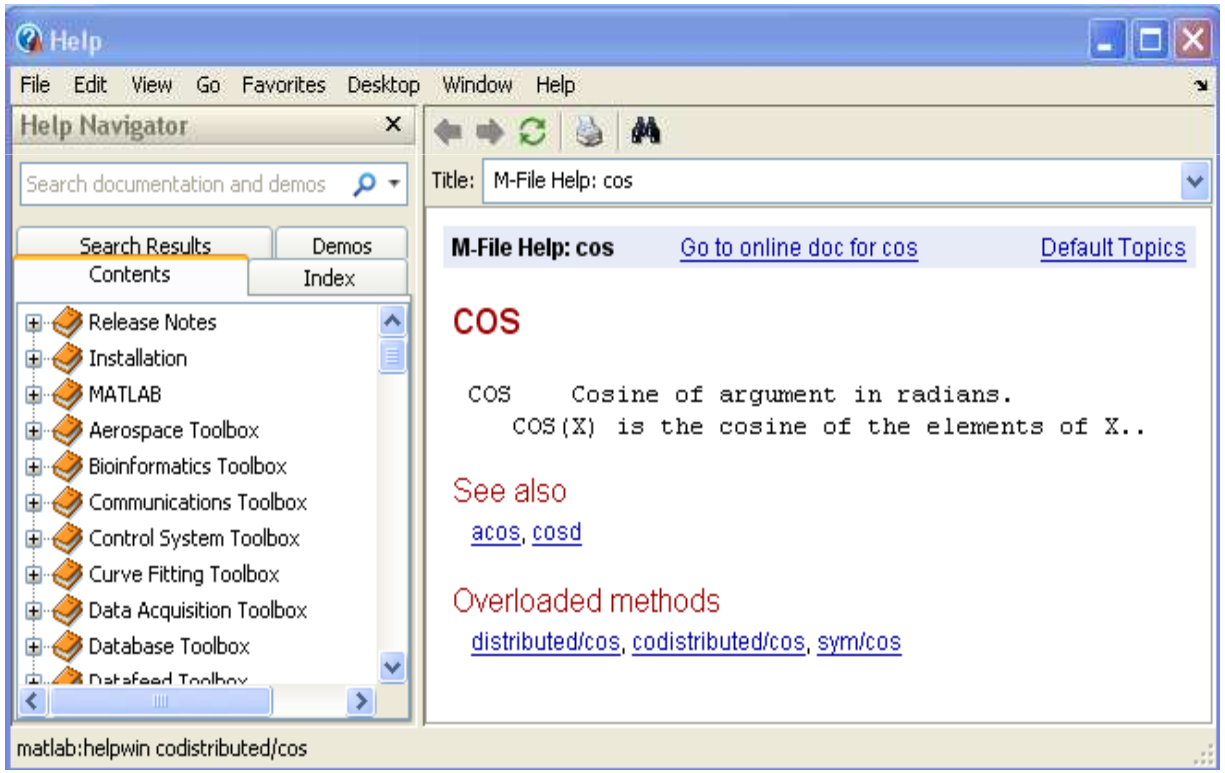
- Desde la línea de comandos

```
>> help cos
COS    Cosine of argument in radians.
   COS(X) is the cosine of the elements of X.

   See also acos, cosd.

   Overloaded methods:
      distributed/cos
      codistributed/cos
      sym/cos

   Reference page in Help browser
      doc cos
```

Ayuda-Help

- Desde la línea de comandos

```
>> helpwin cos
```

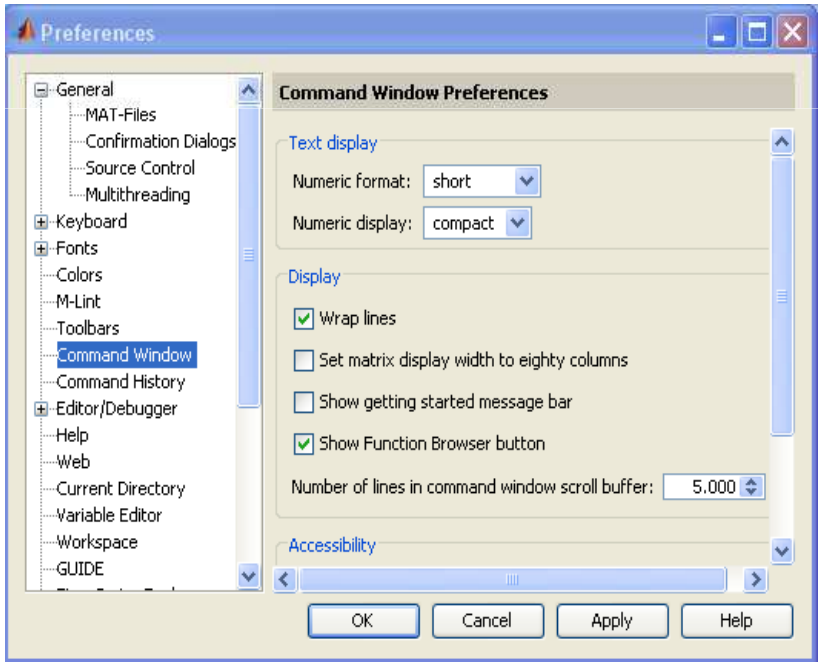


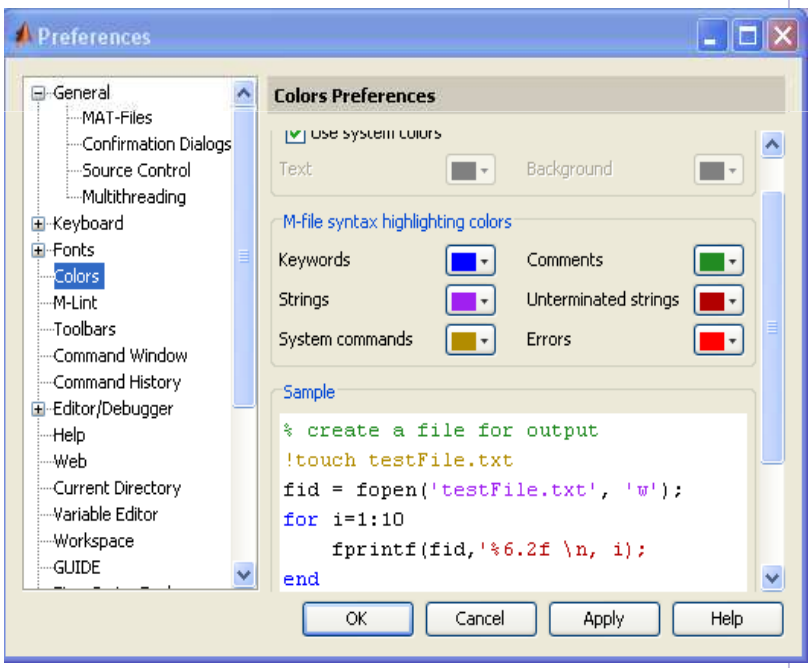
Índice

1. Introducción
2. Entorno de Trabajo
3. Path
4. Ayuda-Help
5. **Preferencias**
6. Formatos de salida
7. Comandos

Preferencias

- Cuadro de diálogo desde el que se establecen casi todas las opciones que el usuario puede determinar por su cuenta.
- *File/Preferences*





Índice

Formatos de salida

- Formatos numéricos con que MATLAB muestra los resultados

short	coma fija con 4 decimales (***defecto***) (41.4527)
long	coma fija con 15 decimales (41.452794857433358)
hex	cifras hexadecimals (4044b9f52e903c94)
bank	números con dos cifras decimales (41.45)
short e	notación científica con 4 decimales (4.1452e+001)
short g	científica o decimal, dependiendo del valor (41.452)
long e	notación científica con 15 decimales (4.145279485743335e+001)
long g	notación científica o decimal, dependiendo del valor (41.4527948574333)
rational	expresa los números racionales como cocientes de enteros

```
>> format long
>> 44/564
ans =
   0.078014184397163
>> format hex
>> 44/564
ans =
   3fb3f8bcd29c2450
```

Índice

Comandos

- SAVE
  - *El comando save, guarda el estado de la sesión de trabajo (por si hay que cerrar Matlab y no se quiere perder ninguna información)*
  - *El comando save filname A x , guarda solo las variables A y x, en un fichero llamado filname*

```
>> save
>> save filname A x
```

- *LOAD*
  - *El comando load, carga el espacio de trabajo que se guardó al cerrar Matlab por última vez.*
  - *El comando load filname , carga las variables guardadas en ese directorio.*

```
>> load
>> load  filname A x
```

Comandos

- *DIARY*
  - Almacena en un fichero un texto que describa lo que el programa va haciendo (la entrada y salida de los comandos utilizados).
  - El comando *diary off* suspende la ejecución de *diary*
  - El comando d*iary on* la reanuda.
  - El simple comando *diary* pasa de *on* a *off* y viceversa.
  - Para poder acceder al fichero *filename.txt* con *Notepad* o *Word* es necesario que *diary* esté en *off*. Si en el comando *diary* no se incluye el nombre del fichero se utiliza por defecto un fichero llamado *diary* (sin extensión).

```
>> diary filename.txt
….
>> diary off
….
>> diary on
```

Comandos

- *DATE*
  - *Devuelve la fecha actual*
- EXIT
  - *Cierra Matlab*
- Control+C
  - *Para la ejecución de un programa*

Otro tipo de datos con MATLAB

Departamento de Ingeniería de Sistemas y Automática
EUITI de Bilbao

Indice

Indice

1. **Cadena de Caracteres**
2. Estructuras
3. Vectores o Matrices de celdas

Cadena de caracteres

❒ MATLAB puede definir variables que contengan cadenas de caracteres (o string).

- Los caracteres de una cadena se almacenan en un vector, con un carácter por elemento.
- Las cadenas de caracteres van entre *apóstrofos* o *comillas simples*.
- Las dos funciones más importantes para el manejo de caracteres son:
  - double(c) convierte en números ASCII cada carácter
  - char(v) convierte un vector de números v en una cadena de caracteres

```
 >> c='motor'
c =
motor
>> v=double(c)    % convierte la cadena en números ASCII
>> char(v)  % convierte números ASCII en caracteres
ans =
motor
```

Indice

1. Cadena de Caracteres
2. **Estructuras**
3. Vectores o Matrices de celdas

Estructuras

- Una estructura (*struct*) es una agrupación de datos de diferente tipo, *bajo un mismo nombre.*
- Estos datos se llaman *miembros* (*members*) o *campos* (*fields*).
- Se pueden crear estructuras por:
  - ❒ Creando cada campo mediante la forma: estructura.campo

```
>> AlumnoFeat.nombre='Miguel‘
AlumnoFeat.=
    nombre: 'Miguel'
>> AlumnoFeat.carnet=73345665
AlumnoFeat. =
    nombre: 'Miguel'
    carnet: 73345665
```

  - ❒ Por medio de la función *struct*. Los *nombres de los campos* se pasan a la función *struct()* entre apóstrofos ('), seguidos del valor que se les quiere dar

```
>> AlumnoFeat=struct('nombre','Mikel','carnet',71233576)
AlumnoFeat=
    nombre: 'Mikel'
    carnet: 71233576
```

Estructuras

- Comandos últimos de Estructura:
  - fieldnames (nombre_estructrua): Nombra los campos que tiene una estructura

```
>>featNombres= fieldnames(AlumnoFeat.)
featNames =
    'nombre'
    ‘carnet'
```

  - rmfield(nombre_estructura, ‘campo’): Elimina el campo elegido de una estructura

```
>>rmfield(AlumnoFeat.,‘carnet')

AlumnoFeat =

    nombre: 'miguel‘
```


Estructuras

- Un vector con diferentes elementos con la misma estructura.
- Crea un vector de 3elementos cada uno de los cuales es una estructura tipo *Alumno*. Sólo el elemento 3del vector es inicializado con los argumentos de la f*unción struct()*.

```
>> AlumnoFeat (3)=struct('nombre', 'Mikel', 'carnet', 71233576)
```

- Para rellenar el resto de elementos:

```
>> AlumnoFeat (2).nombre='noelia';  alum(2).carnet=68845665;
```

- Si después se quiere añadir un nuevo campo y rellenarlo solo en el elemento 5:

```
>> AlumnoFeat (1).edad=18;
```

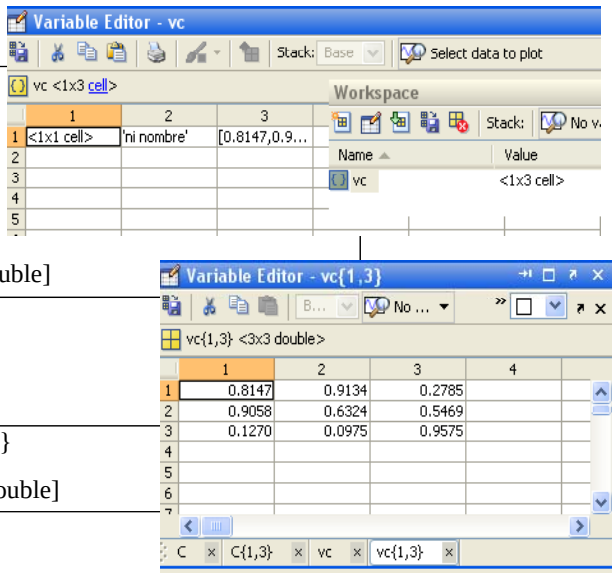
Indice

1. Cadena de Caracteres
2. Estructuras
3. **Vectores o Matrices de celdas**

Vectores o Matrices de Celdas *(Cell Arrays)*

- Un vector (matriz) de celdas es un *vector (matriz) cuyos elementos son cada uno de ellos una variable de tipo cualquiera*.
- Se crea, utilizando *llaves* {}
  - Introduciendo cada elemento

```
>> vc{1}={[1 2 3]}
vc =
    [1x3 double]
>> vc{2}={'ni nombre'}
vc =
    [1x3 double]    'ni nombre'
>> vc{3}={rand(3,3)}
vc =
    [1x3 double]    'ni nombre'    [3x3 double]
```

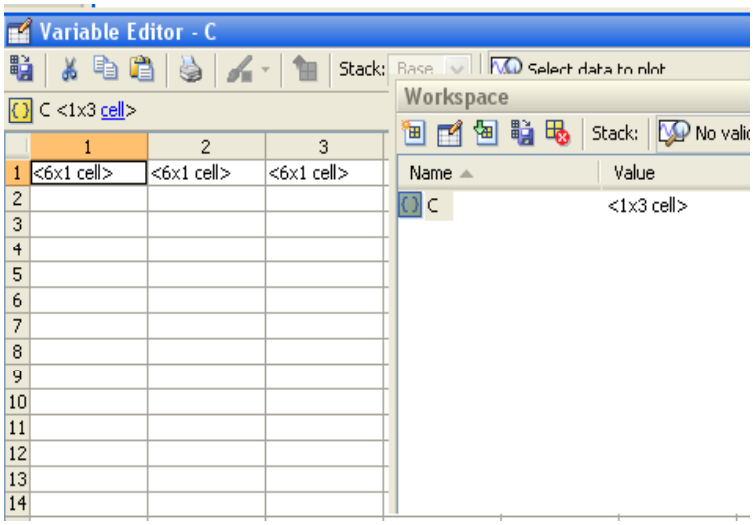


  - Por medio de una sola operación

```
>> vcc={[1 2 3],'mi nombre', rand(3,3)}
vcc =
    [1x3 double]    'mi nombre'    [3x3 double]
```

Vectores o Matrices de Celdas *(Cell Arrays)*

```
>> C{1}={‘Altura';’altura'; ‘impureza%';’ impureza%';'temperatura';'temperatura'}
>> C{2}={'mañana';'noche'; 'mañana';'noche'; 'mañana';'noche'}
>> C{3}={'150';'140'; '85';'95'; '37.2';'36.6‘}
```



Vectores o Matrices de Celdas *(Cell Arrays)*

```
>> C{1}={‘Altura';’altura'; ‘impureza%';’ impureza%';'temperatura';'temperatura'}
>> C{2}={'mañana';'noche'; 'mañana';'noche'; 'mañana';'noche'}
>> C{3}={'150';'140'; '85';'95'; '37.2';'36.6‘}
```

- Acceder a términos del Cell

```
>>  C{1}(2)

ans =

   ‘altura’
```

```
>> C{3}(3)

ans =

   '85'
```

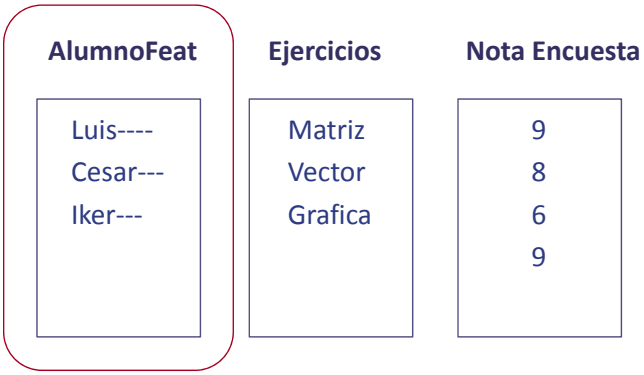
```
>> tanque=  C{1}
 estados
                ‘altura'
                ‘altura'
                ‘impureza%’
               ‘impureza%’
              'temperatura‘
              ‘temperatura‘

>> tanque{2}
               ‘’altura’
```

Ver en el workspace de que tipo es cada variable para saber como acceder a ella

Estructura vs *Cell Arrays*

- Estructura
  - AlumnoFeat
    - **Nombre**
    - **Apellido**
    - **DNI**
    - **Edad**
    - **.....**

- Cell Array
  - NombreCelda



Gráficos

Departamento de Ingeniería de Sistemas y Automática
EUITI de Bilbao

Indice

Indice

Funciones básicas

- Los gráficos 2-D de MATLAB estén fundamentalmente orientados a la representación gráfica de vectores (y matrices).

- Las funciones básicas de los gráficos 2-D son:

plot()	crea un gráfico a partir de vectores y/o columnas de matrices, con escalas lineales sobre ambos ejes
plotyy()	dibuja dos funciones con dos escalas diferentes para las ordenadas, una a la derecha y otra a la izquierda de la figura
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

Indice

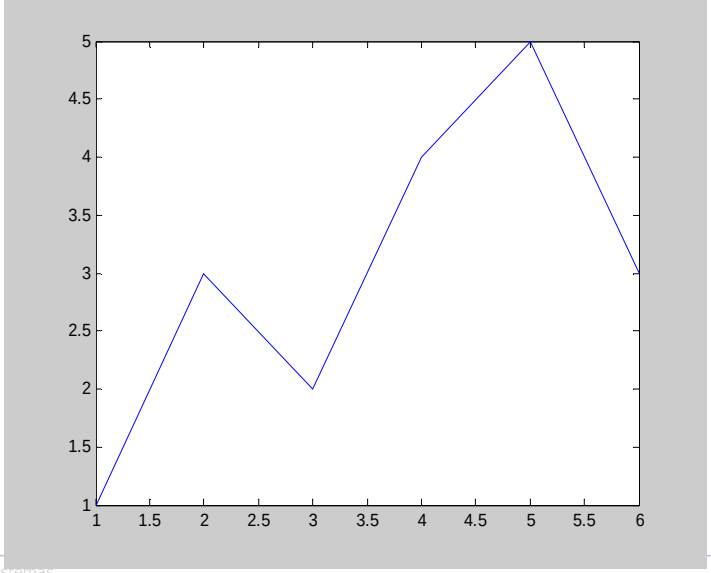
1. Funciones Básicas
2. **Función *Plot()***
3. Marcadores
4. Funciones extras
5. Ventana gráfica

Función *plot()*

- Un único vector como argumento.

índice del vector :abcisas – Valores del vector :ordenadas

```
>> x=[1 3 2 4 5 3];
>> plot(x)
```

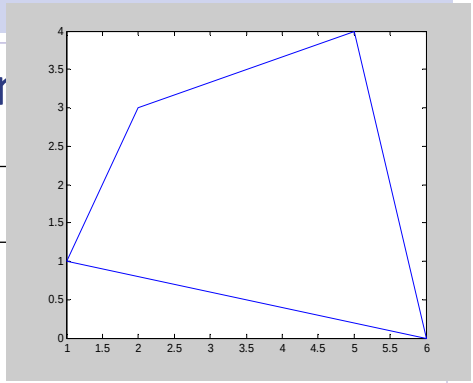


Función *plot()*

- Dos vectores como argumer

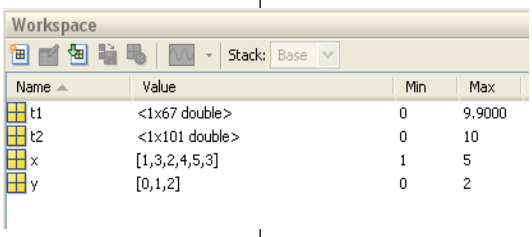
  1º vector: abcisas – 1º vector: ordenadas

```
>> x=[1 6 5 2 1]; y=[1 0 4 3 1];
>> plot(x,y)
```



- IMPORTANTE:
  - la longitud de los vectores debe ser la misma. Mirar en Workspace

```
>> x=[1 2 3 4 5 3]; y=[0 1 2];
>> plot(x,y)
??? Error using ==> plot
Vectors must be the same lengths.
>> lx=length(x), ly=length(y), lt1=length(t1)
lx =
     6
ly =
     3
lt1 =
    67
```



Indice

Marcadores

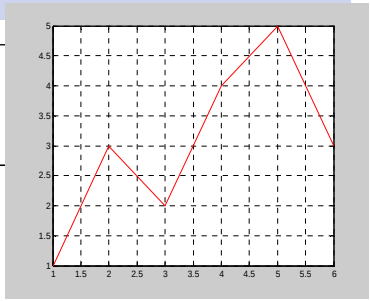
- Dentro de la función *plot*, y entre apostrofes ’ ‘

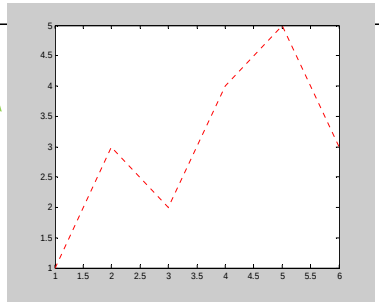
Símbolo	Color	Símbolo	Marcadores (markers)
y	yellow	.	puntos
m	magenta	o	círculos
c	cyan	x	marcas en x
r	red	+	marcas en +
g	green	*	marcas en *
b	blue	s	marcas cuadradas (square)
w	white	d	marcas en diamante (diamond)
k	black	^	triángulo apuntando arriba
		v	triángulo apuntando abajo
Símbolo	**Estilo 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 se seis puntas
--	líneas a trazos		

- *figure()*: Añade una ventana gráfica
- *shg*: muestra la ventana gráfica
- *grid*: añade cuadrícula a la ventana gráfica, para quitarla volver a escribir *grid*.

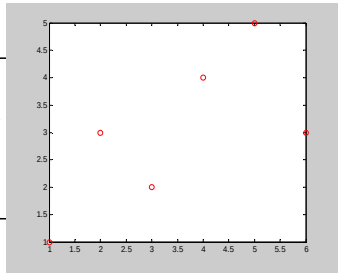
Marcadores

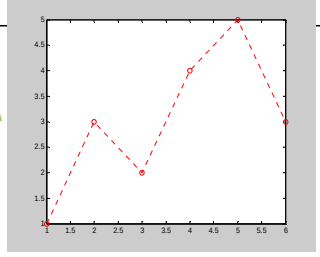
```
>> x=[1 3 2 4 5 3];
>> plot(x,'r'),grid
>> plot(x,'r:')
```





```
>> figure(1)
>> plot(x, 'ro')
>>figure(2)
>> plot(x,'r:o'), shg
```





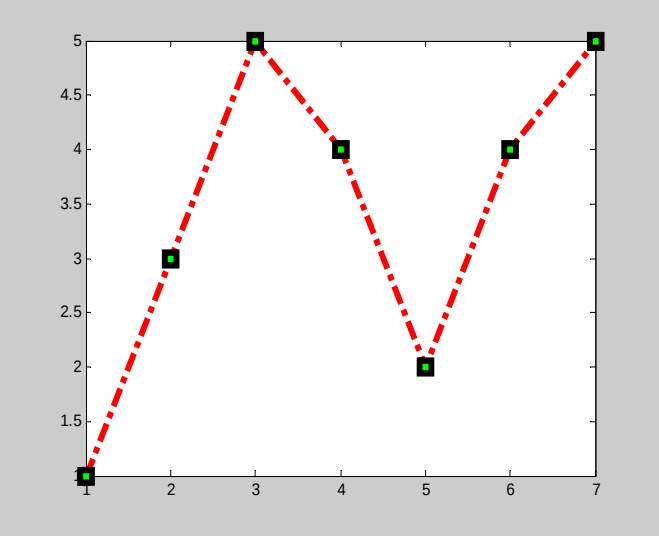
Marcadores

- *Propiedades y valores*
  - A la hora de realizar los gráficos se pueden añadir propiedades y valores más específicos a los marcadores

PROPIEDAD	DESCRIPCIÓN	POSIBLE VALOR DE LA PROPIEDAD
LineWidth	Especifica el color de la línea	Un número representado en unidades de puntos (el valor por defecto es 0.5)
MarkerSize	Especifica el tamaño de las marcas	Un número representado en unidades de puntos
MarkerEdgeColor	Especifica el color del marcador o el color del borde de la línea para marcadores con relleno	Especificadores de color, como los vistos anteriormente, introducidos en forma de cadena
MarkerFaceColor	Especifica el color de relleno de los marcadores	Especificadores de color, como los vistos anteriormente, introducidos en forma de cadena

Marcadores

```
>> x=[1 3 5 4 2 4 5];
>> plot(x,'-.rs', 'LineWidth',4, 'MarkerEdgeColor','k', 'MarkerFaceColor', 'g','MarkerSize',10)
```



ISA – Modelado y Simulación de Sistemas

Funciones extras

1. Funciones Básicas
2. Función *Plot()*
3. Marcadores
4. **Funciones extras**
5. Ventana gráfica

Funciones Gráficas 2D

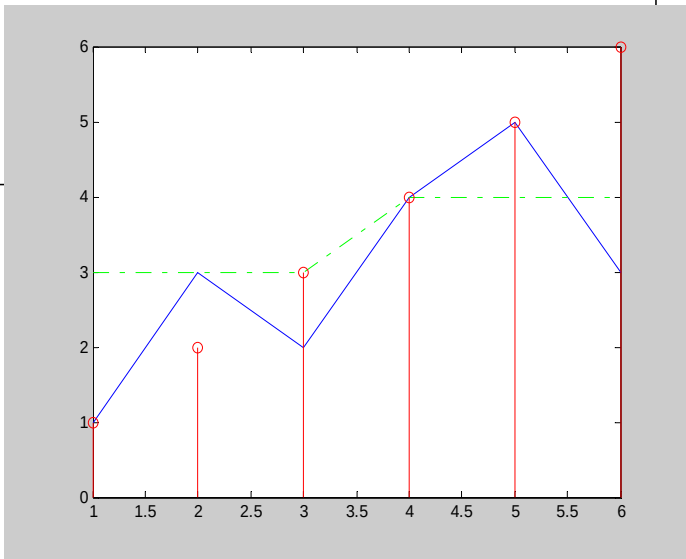
- Función extras:

title('texto')	Añade un título al dibujo
xlabel('texto')	Añade una etiqueta al eje de abscisas. Con xlabel off desaparece
ylabel('texto')	Añade una etiqueta al eje de ordenadas. Con ylabel off desaparece
axis([])	Delimita los ejex axis([xmin,xmax,ymin,ymax])
gtexto('texto')	Introduce texto con ayuda del ratón: el cursos cambia de forma y se espera un clic para introducir el texto en la posición
legend()	Define rótulos para las distintas líneas o ejes utilizados. Para más detalle consultar el Help
grid	Activa la inclusión de una cuadrícula en el dibujo. Con grid off desaparece la cuadrícula
figure	Añade una nueva ventana para la siguiente gráfica
hold on	Añade la siguiente gráfica dentro de la misma ventana. Para desactivar el hold on se utiliza hold off
gold off	Desactiva la función de hold on
close all	Cierra todas las ventanas gráficas abiertas

Funciones extras

- Función *hold on*:
  - En una misma ventana se dibujan varias gráficas.

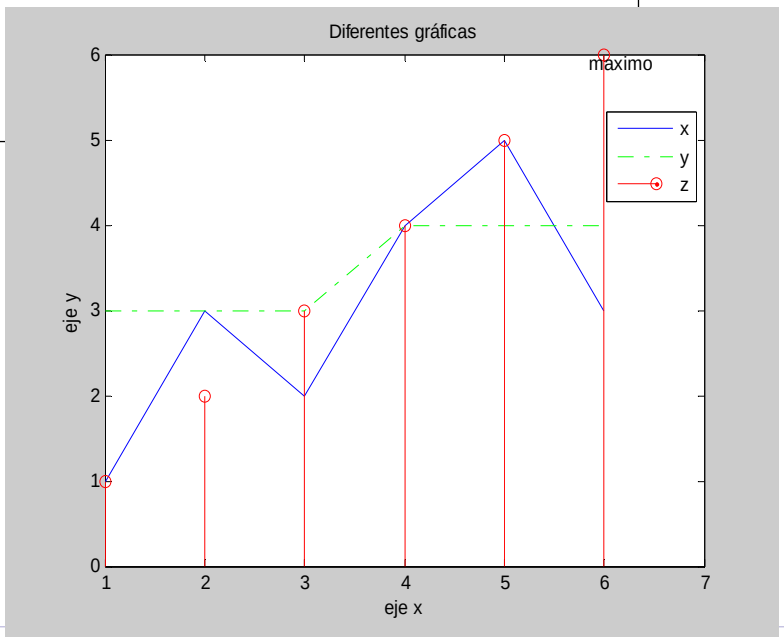
```
>> x=[1 3 5 4 2 4 5];
>> plot(x)
>>hold on
>>y=[3 3 3 4 4 4];
>>plot(y, 'g-. ')
>>z=[1 2 3 4 5 6];
>>stem(z, 'r')
```



Funciones Gráficas 2D

- Función extras:

```
>> title('Diferentes gráficas')
>> xlabel('eje x'); ylabel('eje y')
>>axis([1 7 0 6])
>>y=[3 3 3 4 4 4];
>>gtext ('maximo')
>>legend('x', 'y', 'z')
```

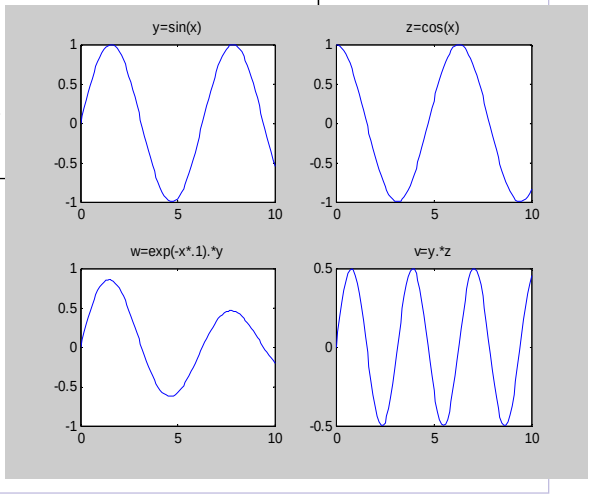


Funciones Gráficas 2D

- Función *subplot()*:
  - Una ventana gráfica se puede dividir en m particiones horizontales (filas) y n verticales (columnas), *i* la posición de la figura a representar .

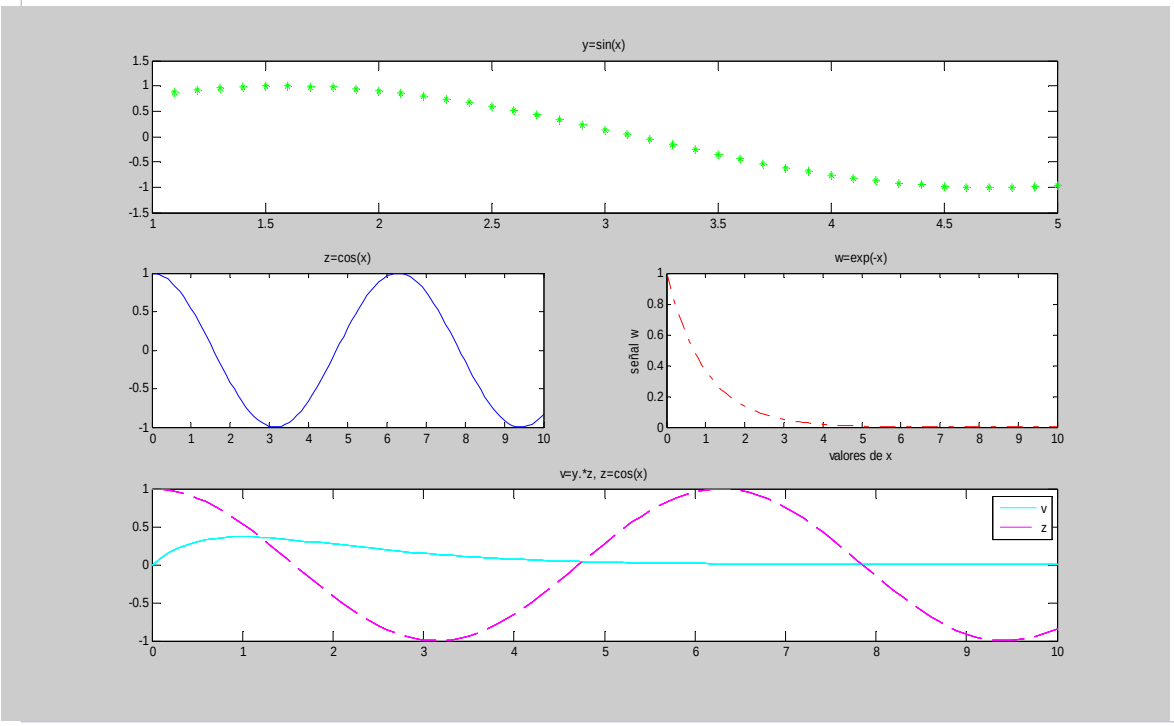
```
>> subplot(m,n,i)
```

```
>> x=0:0.1:10;
>> y=sin(x); z=cos(x); w=exp(-x*.1).*y; v=y.*z
>> subplot(2,2,1), plot(x,y), title('y=sin(x)')
>> subplot(2,2,2), plot(x,z), title('z=cos(x)')
>> subplot(2,2.3), plot(x,w), title('w=exp(-x*.1).*y
>> subplot(2,2,4), plot(x,v), title('v=y.*z')
```



Funciones Gráficas 2D

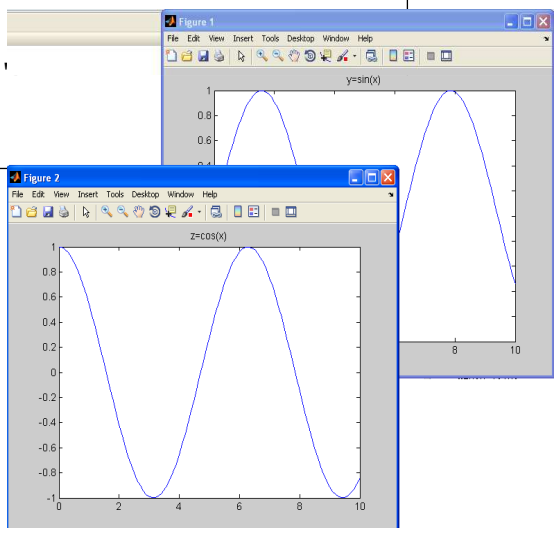
- Realizar la siguiente ventana gráfica, siendo x=0:0.1:10



Funciones Gráficas 2D

- Realizar los 4 gráficos anteriores en distintas ventanas gráficas y cerrarlas todas mediante el comando *close all*

```
>> x=0:0.1:10;
>> y=sin(x); z=cos(x); w=exp(-x*.1).*y; v=y.*z
>> figure; plot(x,y), title('y=sin(x)')
>> figure; plot(x,z), title('z=cos(x)')
>> figure; plot(x,w), title('w=exp(-x*.1).*y '
>> figure; plot(x,v), title('v=y.*z')
>> close all
```



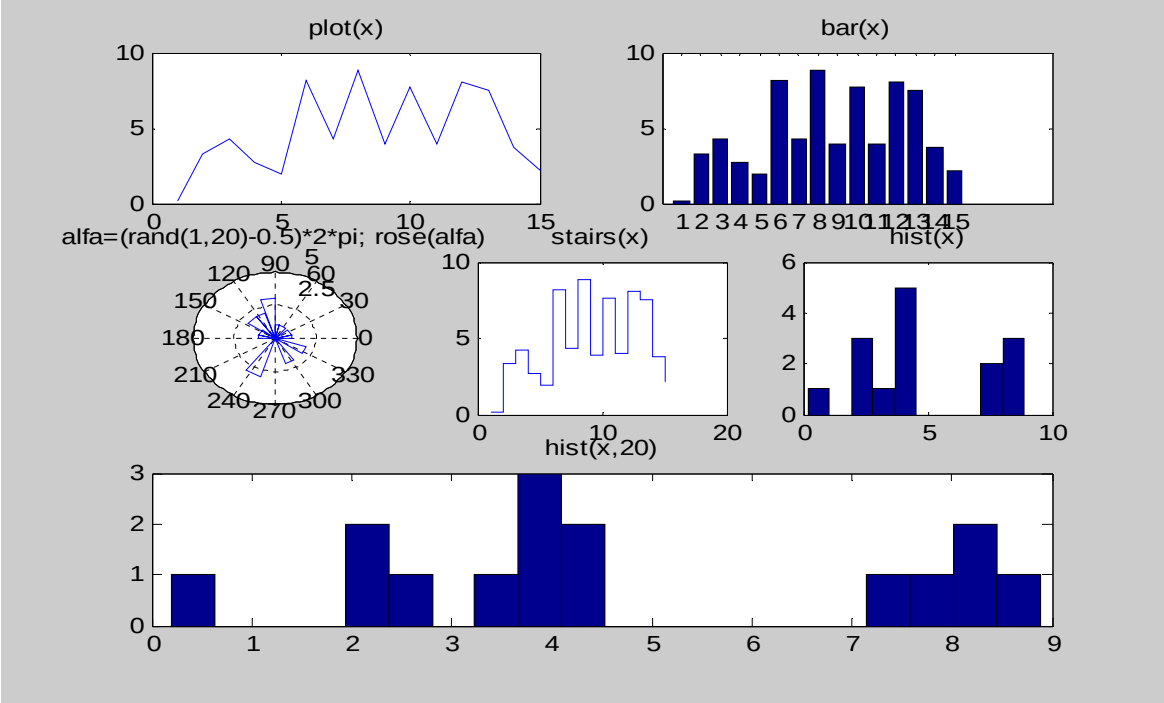
Funciones Gráficas 2D

- OTRAS FUNCIONES GRÁFICAS

bar()	crea diagramas de barras
barh()	diagramas de barras horizontales
bar3()	diagramas de barras con aspecto 3-D
pie()	gráficos con forma de "tarta"
pie3()	gráficos con forma de "tarta" y aspecto 3-D
area()	similar ***plot()***, pero rellenando en ordenadas de 0 a y
stairs()	función análoga a ***bar()*** sin líneas internas
errorbar()	representa sobre una gráfica –mediante barras– valores de errores
compass()	dibuja los elementos de un vector complejo como un conjunto de vectores partiendo de un origen común
feather()	dibuja los elementos de un vector complejo como un conjunto de vectores partiendo de orígenes uniformemente espaciados sobre el eje de abscisas
hist()	dibuja histogramas de un vector
rose()	histograma de ángulos (en radianes)
quiver()	dibujo de campos vectoriales como conjunto de vectores

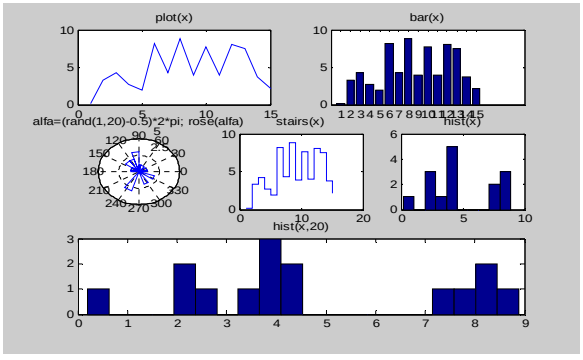
Funciones Gráficas 2D

- OTRAS FUNCIONES GRÁFICAS >> x=[rand(1,15*10)];



Funciones Gráficas 2D

- OTRAS FUNCIONES GRÁFICAS




```
>> x=[rand(1,15)*10];
>> subplot(3,2,1), plot(x), title('plot(x)')
>> subplot(3,2,2), bar(x), title('bar(x)')
>> subplot(3,3,4), alfa=(rand(1,20))*2*pi,; rose(alfa);
     title('alfa=(rand(1,20))*2*pi; rose(alfa)')
>> subplot(3,3,5), stairs(x), title('stairs(x)')
>> subplot(3,3,6), hist(x), title('hist(x)')
>> subplot(3,1,3), hist(x,20), title('hist(x,20)')
```

Funciones Gráficas 2D

- Función *fplot()*:
  - *Si se quiere dibujar una FUNCION, antes de ser pasada por el plot, debe de ser CONVERTIDA a un vector de valores. De eso se encarga la función fplot()*
  - La forma más general de usar la función es la siguiente, donde los campos *cadena* y *tol* son opcionales

```
>> fplot('funcion', limites, 'cadena', tol)
```

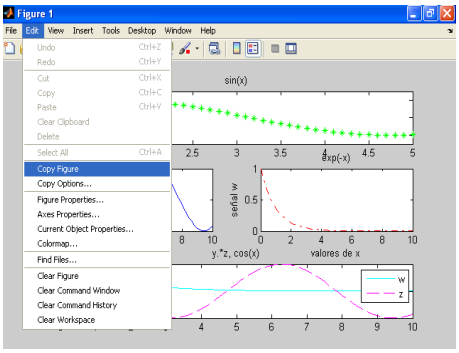
donde:

'funcion'	representa el nombre de la función o del fichero ***.m*** entre apóstrofos (pasado como cadena de caracteres)
limites	es un vector de 2 ó 4 elementos, cuyos valores son [**xmin,xmax**] o [**xmin,xmax,ymin,ymax**]
'cadena'	tiene el mismo significado que en ***plot*** y permite controlar el color, los markers y el tipo de línea
tol	es la tolerancia de error relativo. El valor por defecto es 2e-03. El máximo número de valores en **x** es (1/tol)+1

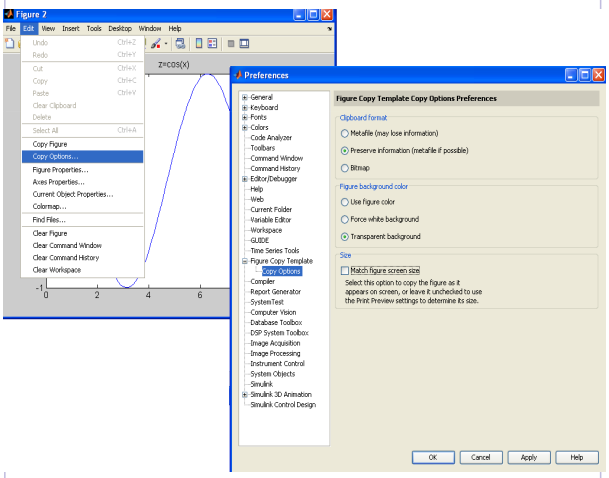
Indice

Ventana gráfica

- Dentro de la ventana gráfica, se pueden acceder a diferentes parámetros
  - Edit – *Copy Figure*: para copiar el gráfico y pegarlo en cualquier editor, etc..

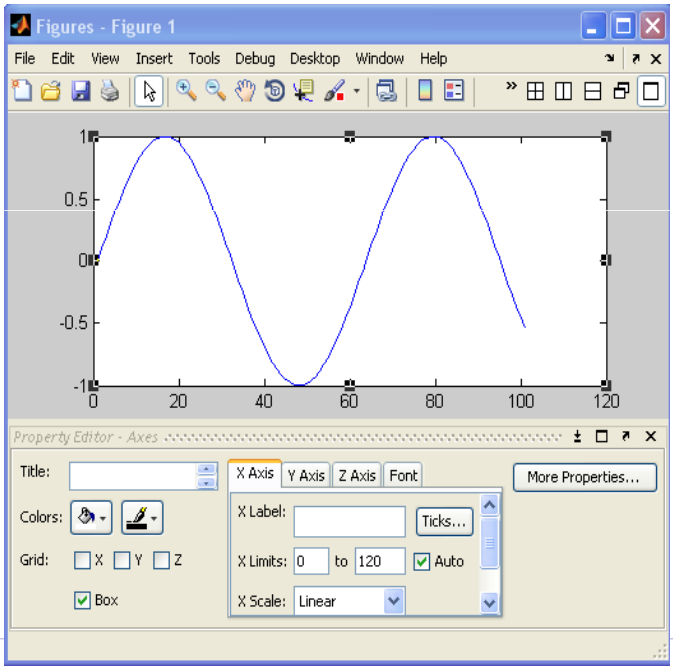


  - Edit – *Copy Option : Si quiere que se copie la figura sobre fondo transparente (sin gris)*



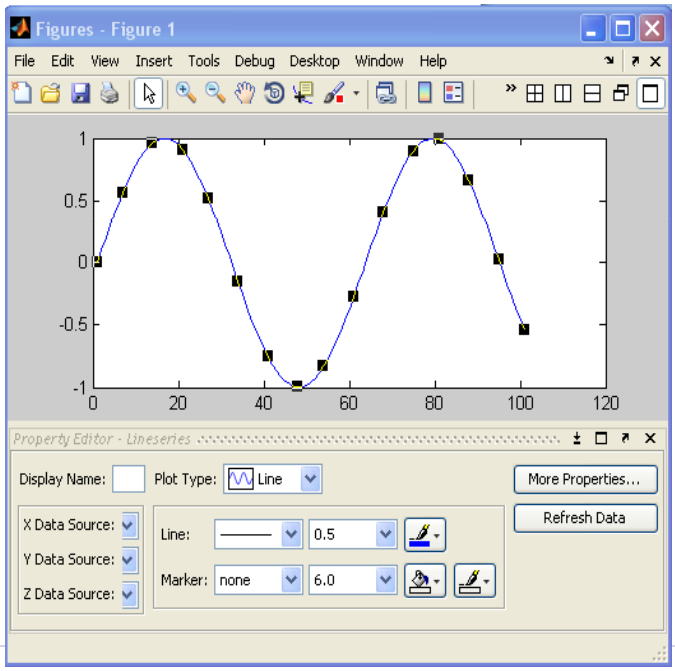
Ventana gráfica

- Dentro de la ventana gráfica, se pueden acceder a diferentes parámetros
  - Edit - *Axes Properties:*



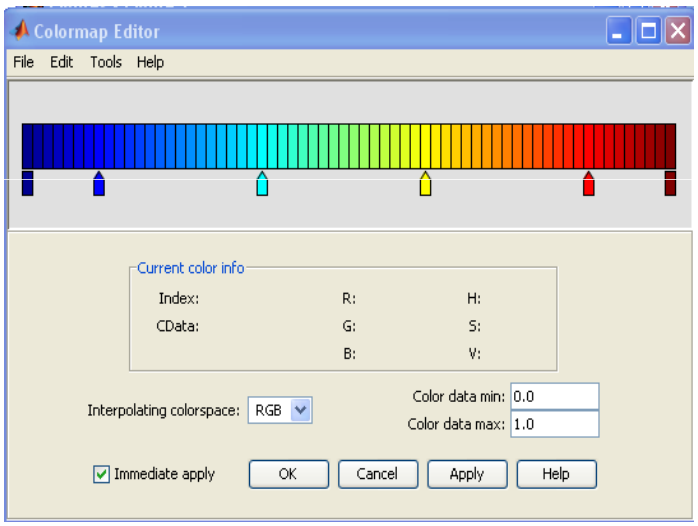
Ventana gráfica

- Dentro de la ventana gráfica, se pueden acceder a diferentes parámetros
  - Edit - Current Objet Properties:



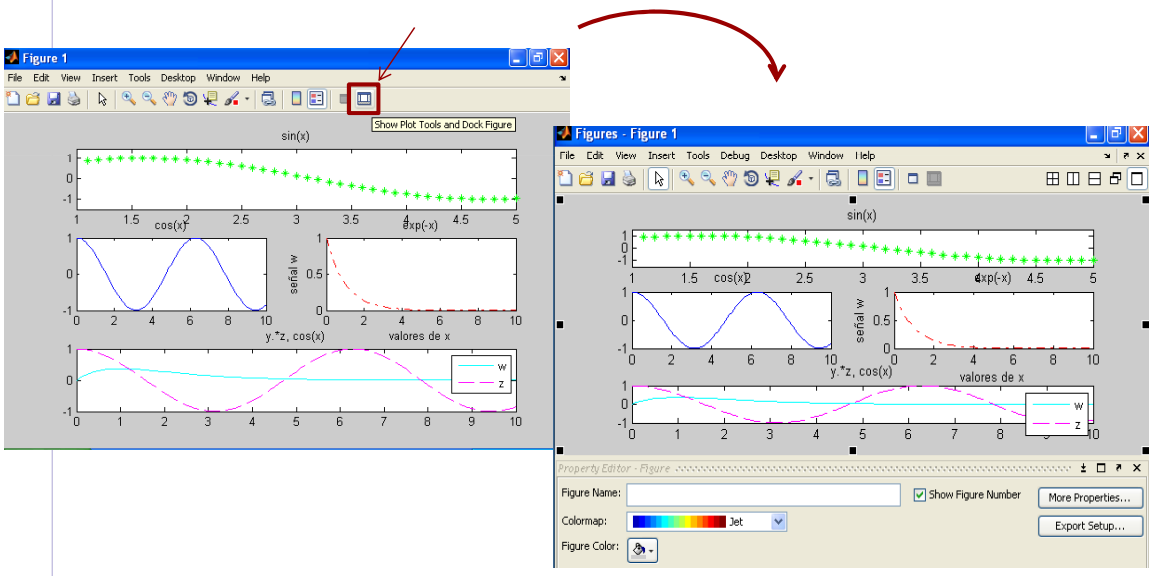
Ventana gráfica

- Dentro de la ventana gráfica, se pueden acceder a diferentes parámetros
  - Edit - *Colormap:*



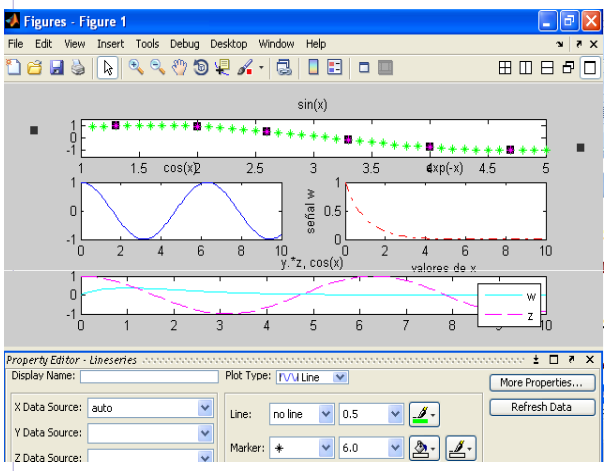
Ventana gráfica

- Se pueden modificar todas las propiedades de la gráfica (texto, título, tamaño de las fuentes, colores etc..) desde la propia ventana gráfica

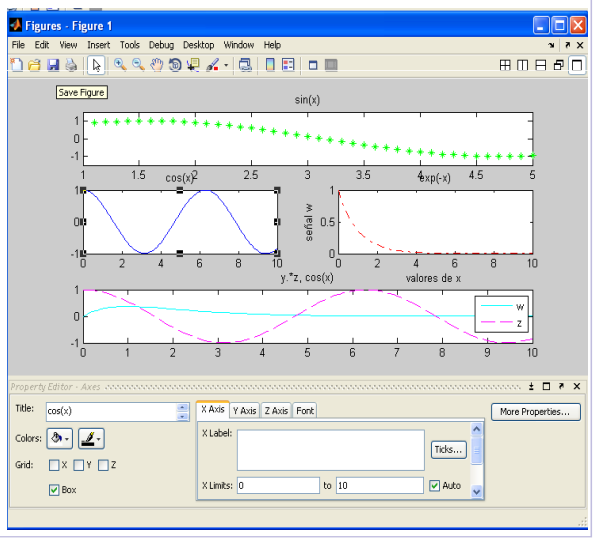


Ventana gráfica

- Para modificar la señal, se pincha en ella

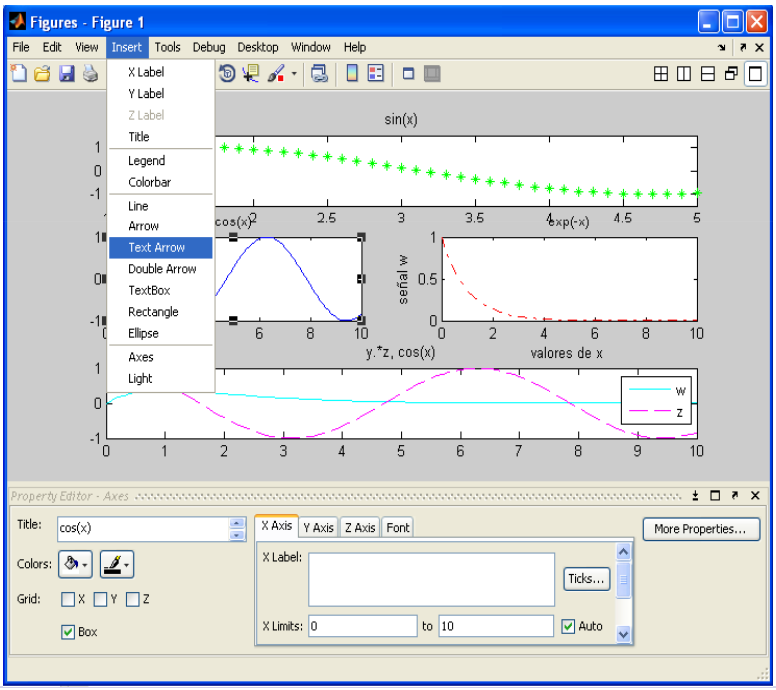


Para modificar los ejes, se pincha en el eje correspondiente



Ventana gráfica

- Se puede insertar un texto, flechas, etc….



Matrices y Vectores

Departamento de Ingeniería de Sistemas y Automática
EUITI de Bilbao

Indice

Indice

1. **Variables y Expresiones**
2. Definición de Matrices desde Teclado
3. Operaciones con Matrices
4. Otra forma de definir Matrices
5. Tipos de datos
6. Operadores

Variables y Expresiones

- Una *variable* es un nombre que se da a una entidad numérica, que puede ser una matriz, un vector o un escalar.
- La forma más normal de cambiar el valor de una variable es colocándola a la izquierda del operador de asignación (=).
- Una expresión de MATLAB puede tener las dos formas siguientes:
  1. Asignando su resultado a una variable: variable = expresión
  2. Evaluando simplemente el resultado del siguiente modo: expresión

```
>>% expresión
>> 15+2-1
ans =
   16
>>% Variable = Expresion
>>x=3*5-12
```

Variables y Expresiones

- El comando *clear* tiene varias formas posibles:
  - clear sin argumentos, *clear* elimina todas las variables creadas previamente (excepto las variables globales).
  - *clear A x y:* Borra las variables indicadas.
  - *clear global :* Borra las variables globales.
  - *clear all*: Borra todas las variables, incluyendo las globales, y las funciones.

- Crear diferentes variables , borrar alguna y comprobar en el Worspace que se han borrado.

Variables y Expresiones

- Para crear un vector FILA. Lo elementos de una fila están separados por espacio o por coma.

```
>> Fila = [1 2 3 4];
>> Fila = [1,2,3,4]
```

- Para crear un vector COLUMNA. Lo elementos de una columna están separados por punto y coma. La Columna es la fila transpuesta.

```
>> Columna = [1; 2; 3; 4];
>> Columna = Fila'
```

Variables y Expresiones

1. Crear un vector fila llamado Par con los cinco primero números pares.
2. Crear un vector columna llamado impar con los cinco primero números impares. (ojo con mayúsculas y minúsculas).
3. Crear una matriz llamada A, con los elementos de la primer fila: 1 2 3 y la segunda: 4 5 6.
4. Comprobar en el worsapce y array editor las variables creadas
5. Borrar SOLO la variable impar
6. Comprobar en el worksapce el resultado
7. Borrar todas las variables
8. Comprar en el workspace el resultado

Indice

1. Variables y Expresiones
2. **Definición de Matrices desde Teclado**
3. Operaciones con Matrices
4. Otra forma de definir Matrices
5. Tipos de datos
6. Operadores

Definición de Matrices desde Teclado

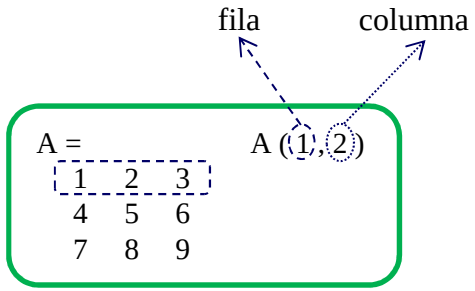
- Definir matrices
  - Para definir una matriz no hace falta declararla o establecer de antemano su tamaño.
  - Las matrices se introducen por filas, se *almacenan por columnas.*
  - Las filas están separadas (;), y los elementos de cada fila espacio o (,).

- Acceder a un elemento de la matriz
  - Se accede a los elementos de un vector poniendo el índice entre paréntesis x(3).
  - Se accede a los elementos de una matriz :
    - Poniendo los índices separados por una coma (fija,columa) A(1,2).
    - *Mediante un índice (almacenado por columna)*

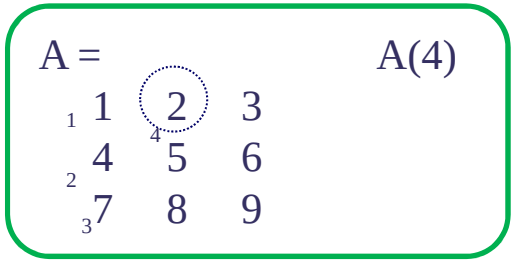
Definición de Matrices desde Teclado

- Definir matrices la matriz A con la primera fila= 1 2 3 , la segunda fila=4 5 6, la tercera fila=7 8 9.
- Acceder al segundo elemento de la primera fila

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



Almacenamiento columna



Definición de Matrices desde Teclado

- Se ha tomado la tensión (tensión alta y tensión baja) a 6 pacientes en la sala de curas del hospital. Crear una matriz con las tensiones alta y bajas de los 6 pacientes. Cada paciente es una columna, los valores de la tensión alta están en la primera fila y la baja en la segunda.

  P1 = 100, 72 P4 = 120, 95

  P2 = 132, 85 P5 = 116, 88

  P3 = 92, 66 P6 = 160, 100

- Acceder a la tensión baja del cuarto paciente.

Definición de Matrices desde Teclado

- A esos mismos 6 pacientes se le tomó la temperatura 37.6, 36.4, 35.9, 36.5, 38.3, 39.1. Crear un vector con la temperatura los pacientes y acceder a la temperatura del tercer paciente

Indice

1. Variables y Expresiones
2. Definición de Matrices desde Teclado
3. **Operaciones con Matrices**
4. Otra forma de definir Matrices
5. Tipos de datos
6. Operadores

Operaciones con matrices

- Operadores matriciales

+	adicción o suma
–	sustracción o resta
*	multiplicación
'	traspuesta
^	potenciación
\	división-izquierda
/	división-derecha

```
>> [1 2 3 4]^2
??? Error using ==> mpower
Matrix must be square.
>> [1 2 3 4].^2
ans =
     1     4     9    16
```

- Operadores elemento a elemento

.*	producto elemento a elemento
./ y .\	división elemento a elemento
.^	elevar a una potencia elemento a elemento

Operaciones con matrices

1. Volver a crear el vector Par, el impar y la matriz A
2. Transponer el vector impar
3. Crear una matriz B (2x3), siendo la primer fila todo 1, y la segunda 2.
4. Sumar la matriz A y la B.
5. Multiplicar el vector Par y el impar.
6. Multiplicar elemento a elemento el vector Par y el impar.

Indice

1. Variables y Expresiones
2. Definición de Matrices desde Teclado
3. Operaciones con Matrices
4. **Otra forma de definir Matrices**
5. Tipos de datos
6. Operadores

Otra forma de definir Matrices

- MATRICES PREDEFINIDAS
  - eye(4) Forma la matriz unidad de tamaño (4×4)
  - zeros(3,5) forma una matriz de ceros de tamaño (3×5)
  - zeros(4) ídem de tamaño (4×4)
  - ones(3) forma una matriz de unos de tamaño (3×3)
  - ones(2,4) idem de tamaño (2×4)
  - magic(4) crea una matriz (4×4) con los números 1, 2, ... 4*4, con la propiedad de que todas las filas y columnas suman lo mismo
  - rand(3) forma una matriz de números aleatorios entre 0 y 1, con distribución uniforme, de tamaño (3×3)
  - Rand(1,6) forma un vector fila de 6 columnas
  - n=length(x) calcula el número de elementos de un vector x
  - [m,n]=size(A) devuelve el número de filas y columnas de la matriz A
  - zeros(size(A)) forma una matriz de ceros del mismo tamaño quede una matriz A previamente creada

Otra forma de definir Matrices

1. Crear una vector fila “x” de 6 elementos todos 1
2. Crear una matriz cualquiera “A” de 2x2 (con rand)
3. Escribir el vector y=[1 3 5 7 9 11 13 15 17 19] y calcular la longitud de ese vector
4. Crear una matriz de ceros “Z” del mismo tamaño que la matriz “A”

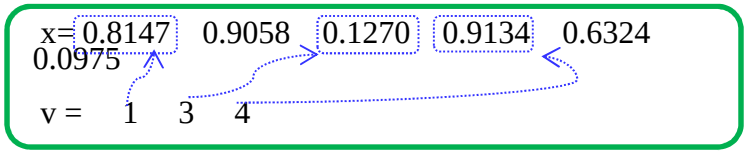
Otra forma de definir Matrices

1. Crear la matriz A=[1 2 3 4;5 6 7 8] y la B=[1 2 3;4 5 6;7 8 9].
2. Crear otra matriz C de ceros, que tenga el numero de filas como el numero de filas de A y el numero de columnas de B.

Otra forma de definir Matrices

- DIRECCIONAMIENTO
  - Direccionamiento de un vector mediante otro vector.

```
>> v=[1 3 4]
v =
    1    3    4
>> x=rand(1,6)
x =
    0.8147    0.9058    0.1270    0.9134    0.6324
0.0975
>> x(v)
    0.8147    0.1270    0.9134
```



  - Direccionamiento de una matriz mediante un vector.

```
>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> f=[1 7  5  13];
>> A(f), A(5), A(6)
ans =
    16     7     2     8
ans =     2
ans =    11
```

Otra forma de definir Matrices

- DIRECCIONAMIENTO
  - Direccionamiento de un vector mediante otro vector.

```
>> v=[1 3 4]
v =
    1    3    4
>> x=rand(1,6)
x =
    0.8147    0.9058    0.1270    0.9134    0.6324
0.0975
>> x(v)
    0.8147    0.1270    0.9134
```

  - Direccionamiento de una matriz mediante un vector.

```
>> A=magic(4)
A =
   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
>> f=[1 7  5  13];
>> A(f), A(5), A(6)
ans =
   16    7    2    8
ans =     2
ans =    11
```

A =

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

f=[1 7 5 13]

A(f)= 16 7 2 8

Otra forma de definir Matrices

- OPERADOR (:)
  - Este operador es muy importante en MATLAB y puede usarse de varias formas.
  - En cierta forma se podría decir que el *operador (:)* representa un *rango*:
    - Incremento igual a 1:

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

    - Incremento distinto a 1:

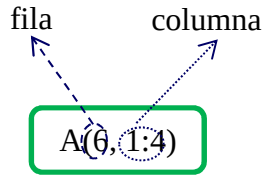
Inicio : incremento : final

```
>> x=1:2:10
x =
     1     3     5     7     9
>> x=1:1.5:10
x =
    1.0000    2.5000    4.0000    5.5000    7.0000    8.5000   10.0000
 >> x=10:-1:1
x =
    10     9     8     7     6     5     4     3     2     1
```

Otra forma de definir Matrices

- OPERADOR (:)
  - Extraer los 4 primeros elementos de la 6ª fila:

```
>> A=magic(6)
A =
   35    1    6   26   19   24
    3   32    7   21   23   25
   31    9    2   22   27   20
    8   28   33   17   10   15
   30    5   34   12   14   16
    4   36   29   13   18   11
>> A(6, 1:4)
ans =
    4   36   29   13
```



  - Los dos puntos aislados representan "todos los elementos". Extraer todos los elementos de la 3ª fila:

```
>> A(3,:)
ans =
   31    9    2   22   27   20
```

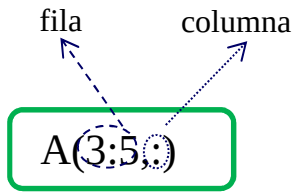
  - Para acceder a la última fila o columna puede utilizarse la palabra *end.*

```
>> A(end,:)
ans =
    4   36   29   13   18   11
```

Otra forma de definir Matrices

- OPERADOR (:)
  - Extraer los conjuntos de filas, de la 3 a la 5 y la 1, 2 y 5

```
>> A(3:5,:)
ans =
   31    9    2   22   27   20
    8   28   33   17   10   15
   30    5   34   12   14   16
>> A([1 2 5],:)
ans =
   35    1    6   26   19   24
    3   32    7   21   23   25
   30    5   34   12   14   16
```



  - Crear una matriz a partir de otra

```
>> B=eye(size(A));
>> B([2 4 5],:)=A(1:3,:)
B =
    1    0    0    0    0    0
   35    1    6   26   19   24
    0    0    1    0    0    0
    3   32    7   21   23   25
   31    9    2   22   27   20
    0    0    0    0    0    1
```

Otra forma de definir Matrices

- MATRIZ VACIA, BORRADO DE FILAS O COLUMNAS []
  - El operador [], crea una matriz vacía o borra una fila o columna

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> A(:,3)=[]
A =
     8     1
     3     5
     4     9
```

Otra forma de definir Matrices

- Calcular los valores del vector "y" sabiendo que la función y=2*x+1 siendo "x" es un vector fila :
  1. Con los 5 primeros números impares
  2. Cuyos 9 primeros números van del 1 al 9 y los otros 9 siguientes son ceros (utilizar la matriz zeros()).
  3. Con los tres últimos números de la quinta fila de la matriz A=rand(6)

Otra forma de definir Matrices

- Dada la matriz :

A=[1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15; 16 17 18 19 20; 21 22 23 24 25].

Crear una matriz B cuyas filas 1 y 3 sean iguales que las filas 1 y 3 de A y las filas 2 4 y 5 sean como las filas de la 1 a la 3 de la matriz A. Después borrar la última columna de la matriz B.

```
A =
    1    2    3    4    5
    6    7    8    9   10
   11   12   13   14   15
   16   17   18   19   20
   21   22   23   24   25
```

```
B =
    1    2    3    4
    1    2    3    4
   11   12   13   14
    6    7    8    9
   11   12   13   14
```

Indice

1. Variables y Expresiones
2. Definición de Matrices desde Teclado
3. Operaciones con Matrices
4. Otra forma de definir Matrices
5. **Tipos de datos**
6. Operadores

Tipos de datos

- TIPOS DE DATOS DE MATLAB
  - Números reales
  - Números complejos
  - Variables Lógicas
  - Cadena de Caracteres

Tipos de datos

- NUMEROS REALES DE DOBLE PRECISIÓN
  - MATLAB trabaja siempre en *doble precisión*, es decir guardando cada dato en 8 bytes, con unas 15 cifras decimales exactas.
  - Casos especiales
    - Números muy grandes Inf (Infinito)

```
>> 1.0/0.0
ans =
   Inf
```

    - Números muy pequeños (NaN Not a Number)

```
>> 0/0
ans =
   NaN
>> inf/inf
ans =
   NaN
```

Tipos de datos

- NUMEROS COMPLEJOS
  - MATLAB trabaja con números complejos.

```
>> a=sqrt(-4)
a =
        0 + 2.0000i
>> 3+4j
ans =
   3.0000 + 4.0000i
```

  - El comando *complex*, crea un número complejo.

```
>> complex(1,2)
ans =
   1.0000 + 2.0000i
```

Tipos de datos

- VARIABLES LOGICAL
  - Variables que sólo pueden tomar los valores *true* (1) y *false* (0). (resultado de una comparación…)

```
>> x=[1 4 6 77 4 8];
>> m=x<5
m =
    1     1     0     0     1     0
>> x(m)=-10 % solo los elementos donde m=1 TRUE
es=10
x =
  -10   -10     6    77   -10     8
```

- CADENA DE CARACTERES
  - MATLAB puede definir variables que contengan cadenas de caracteres. Las cadenas de texto van entre apóstrofos o comillas simples.

```
>> s='lunes'
s =
lunes
```

Indice

1. Variables y Expresiones
2. Definición de Matrices desde Teclado
3. Operaciones con Matrices
4. Otra forma de definir Matrices
5. Tipos de datos
6. **Operadores**

Operadores

- OPERADORES RELACIONALES
  - Matlab dispone de los siguiente operadores relacionales:

<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
==	Igual que
~=	Distinto que

~
Tecla: alt126

  - Al igual que en C, si una comparación se cumple el resultado es 1 (*true*), mientras que si no se cumple es 0 (*false*).

```
>> A=[1 2;0 3]; B=[4 2;1
5];
>> A==B
ans =
     0     1
     0     0
>> A~=B
ans =
     1     0
     1     1
>> 5>8
ans =
     0
```

```
A =

     1     2
     0     3
```

```
B =

     4     2
     1     5
```

Operadores

- OPERADORES RELACIONALES
  - Matlab dispone de algunas funciones para trabajar con operadores relacionales como por ejemplo:
  - *find(x)* busca índices correspondientes a elementos de vectores que cumplen una determinada condición. El resultado es un vector con los índices de los elementos que cumplen la condición
  - *find(A)* cuando esta función se aplica a una matriz la considera como un vector con una columna detrás de otra, de la 1ª a la última

Operadores

- OPERADORES RELACIONALES
  - –Los número de la matriz A mayores que 4, convertirlos en 10

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> m=find(A>4) % devuelve el índice de los elementos
que cumple la condición (A>4)
m =
     1
     5
     6
     7
     8
>> A(m)=10
A =
    10     1    10
     3    10    10
     4    10     2
```

Tipos de datos

- Las notas que 10 alumnos sacaron en un examen son las siguientes:

a1=7.3, a2=5.6, a3=3.2, a4=9.4, a5=4.9, a6=2.3, a7=8.8, a8=8.3, a9=6.4, a10=9.3. Crear un vector con las notas de los alumnos, y a los que tengan más de un 9 pasarles la nota a un 10.

Operadores

- OPERADORES LÓGICOS
  - Matlab dispone de los siguiente operadores lógicos:

&	*and* (función equivalente: **and(A,B)**). Se evalúan siempre ambos operandos, y el resultado es ***true*** sólo si ambos son ***true***.
&&	*and* breve: si el primer operando es ***false*** ya no se evalúa el segundo, pues el resultado final ya no puede ser más que ***false***.
|	*or* (función equivalente: **or(A,B)**). Se evalúan siempre ambos operandos, y el resultado es ***false*** sólo si ambos son ***false***.
||	*or* breve: si el primer operando es ***true*** ya no se evalúa el segundo, pues el resultado final no puede ser más que ***true***.
~	*negación lógica* (función equivalente: **not(A)**)
xor(A,B)	*negación lógica* (función equivalente: **not(A)**)

Negación lógica ~
Tecla: alt126

  - Funcionan con números. Un número distinto de 0 es verdadero y cero falso.
  - Se pueden introducir en expresiones matemáticas y en comandos de flujo de programa (comando if)

Operadores

- OPERADORES LÓGICOS

```
>> x=[9 3 0 11 0 15]; y=[2 0 13 -11 0 4];
>> x&y
ans =
     1     0     0     1     0     1
>> x|y
ans =
     1     1     1     1     0     1
```

Operadores

- Obtener los valores de una función y(x) para valores de x=0:0.5:3. y(x)=

$$\begin{cases} 2 & \text{si } x < 1.5 \text{ o } 2.5 \leq x \\ 1 & \text{si } -1.5 \leq x < 2.5 \end{cases}$$