

INFORMATICA APLICADA A LA INGENIERÍA QUÍMICA: 2ª parte MATLAB

1^{er} Curso del Grado de Ingeniero Químico

Departamento de Ingeniería Química

Universidad Complutense de Madrid

MATLAB es el nombre abreviado de MATrix LABoratory

ES UN PROGRAMA PARA REALIZAR CALCULOS NUMERICOS CON
CUALQUIER TIPO DE NUMERO, CON VECTORES Y MATRICES

TIENE UN LENGUAJE DE PROGRAMACION PROPIO



INFORMATICA APLICADA A LA INGENIERÍA QUIMICA: 2ª parte MATLAB

Razones por las que Matlab se ha convertido en una herramienta tan popular dentro de las Ciencias e Ingenierías:

- Su capacidad de realizar operaciones con matrices de forma sencilla. Pueden realizarse operaciones con matrices sin necesidad de preocuparse por programar bucles que recorran los elementos de las matrices.
- Ni siquiera es necesario definir las dimensiones de las matrices utilizadas. Además estas dimensiones pueden alterarse si es necesario.
- Presenta una gran facilidad de crear gráficos sencillos, e interfaces de usuario
- Existen un gran número de bibliotecas adicionales de funciones (toolboxes)

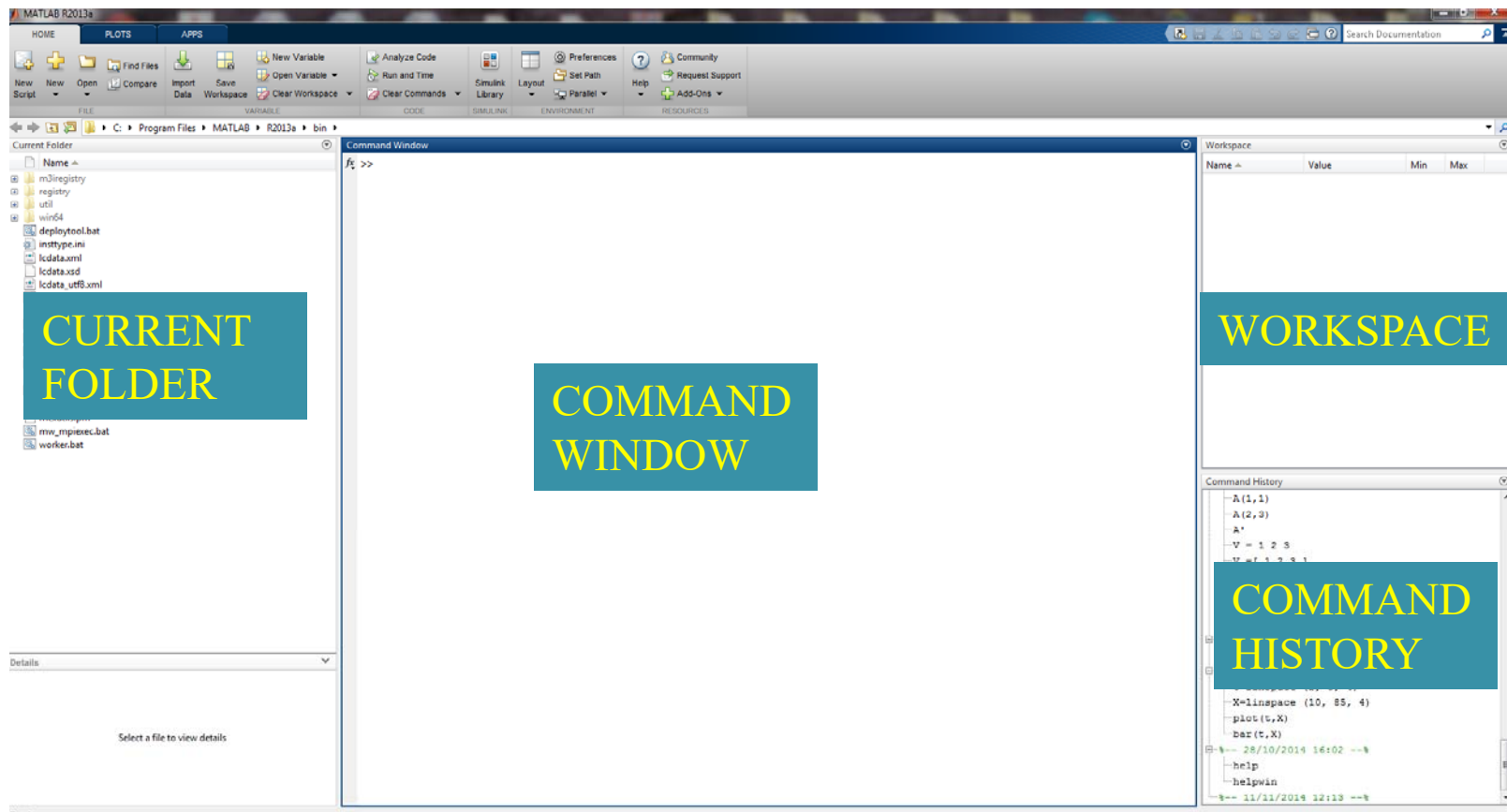


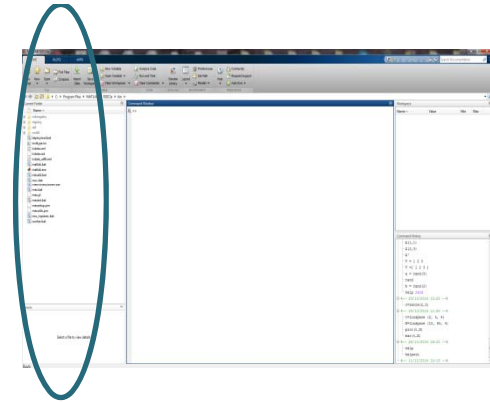
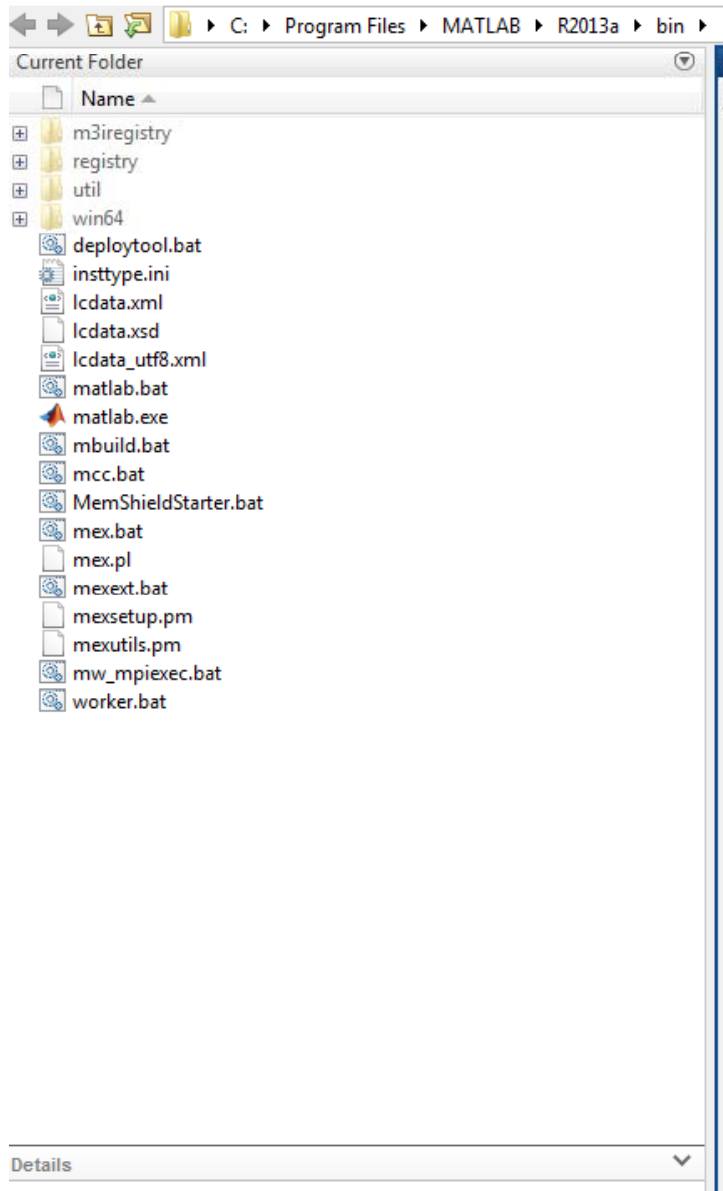
INFORMATICA APLICADA A LA INGENIERÍA QUIMICA: 2ª parte MATLAB

Tema 1: Conceptos básicos de MATLAB

MATLAB Desktop = el escritorio de MATLAB.

Es la ventana más general de la aplicación; el resto de las ventanas pueden alojarse en ella, o ejecutarse como ventanas independientes.





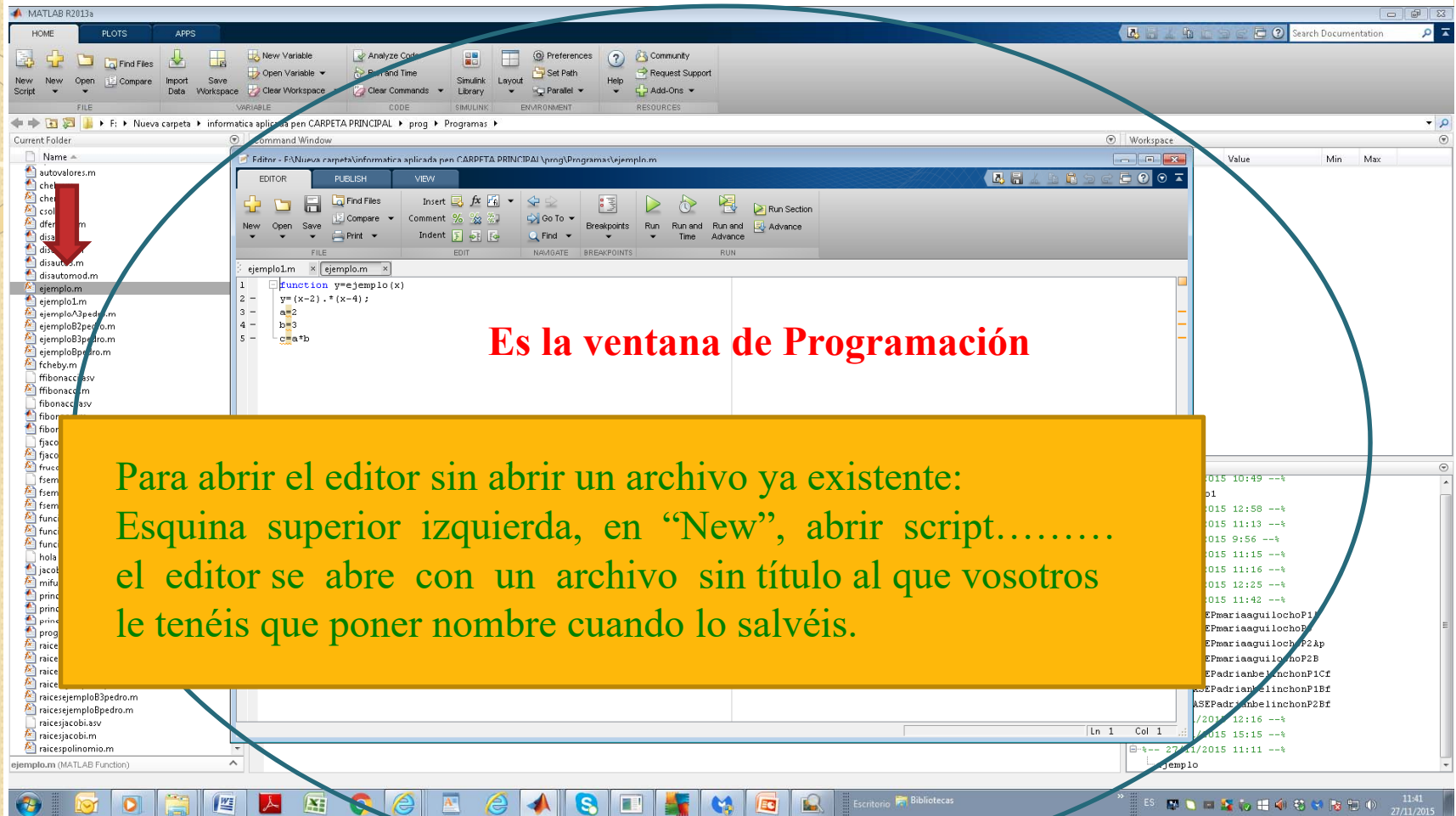
- **Current folder (o current directory):**
Muestra los ficheros del directorio activo o actual.

Clicando dos veces sobre alguno de los ficheros *.m del directorio activo se abre el **editor de ficheros** de MATLAB, herramienta fundamental para la programación.

LA EXTENSIÓN ADECUADA ES .m

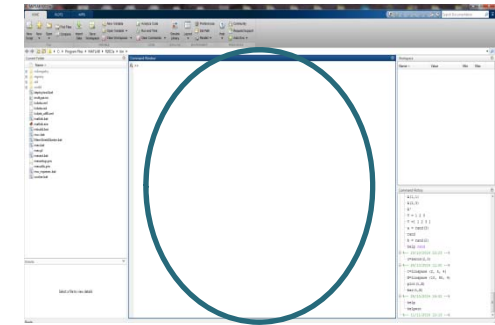
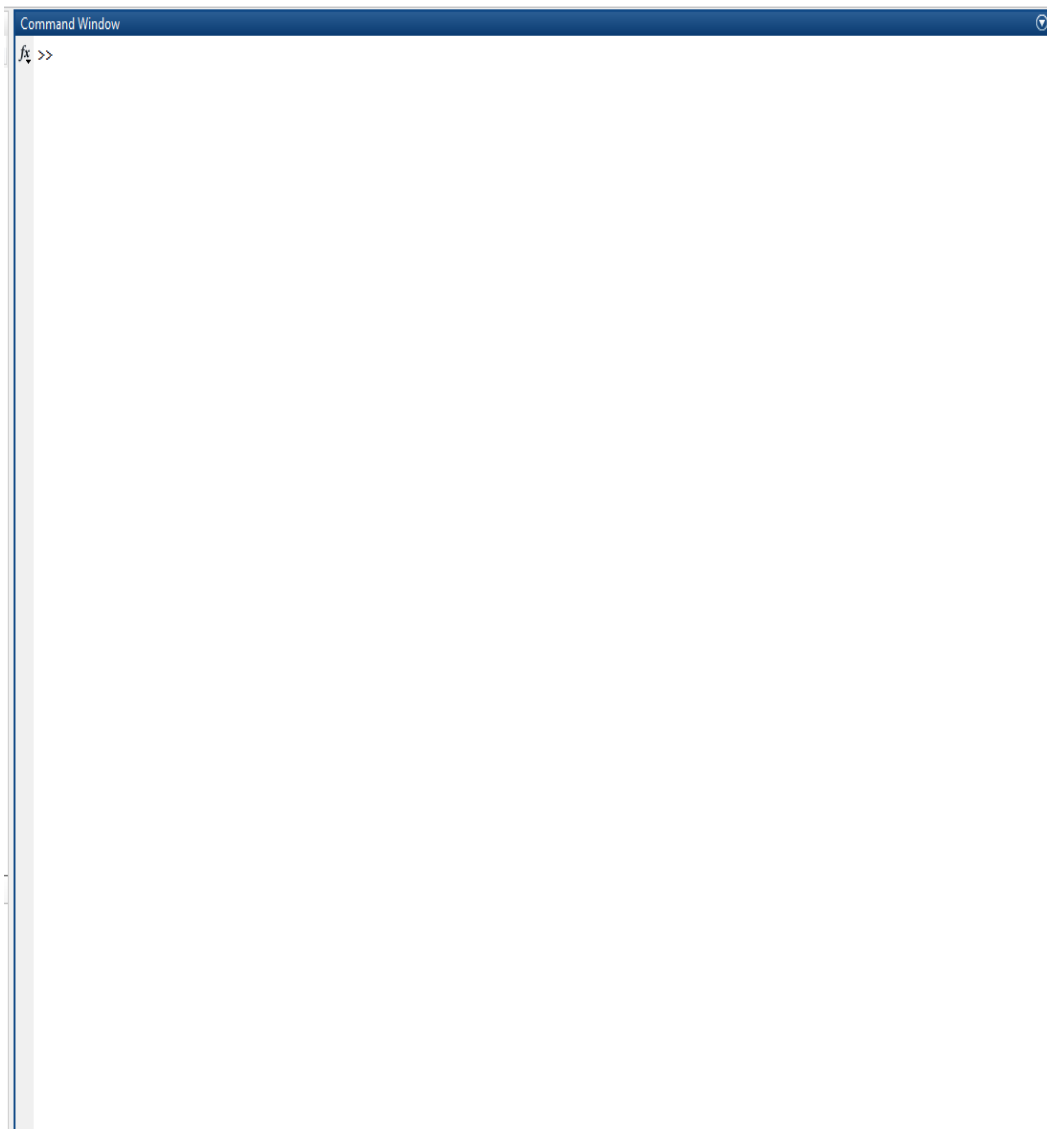
Nunca .mat

Editor de ficheros de MATLAB, herramienta para la programación.



Es la ventana de Programación

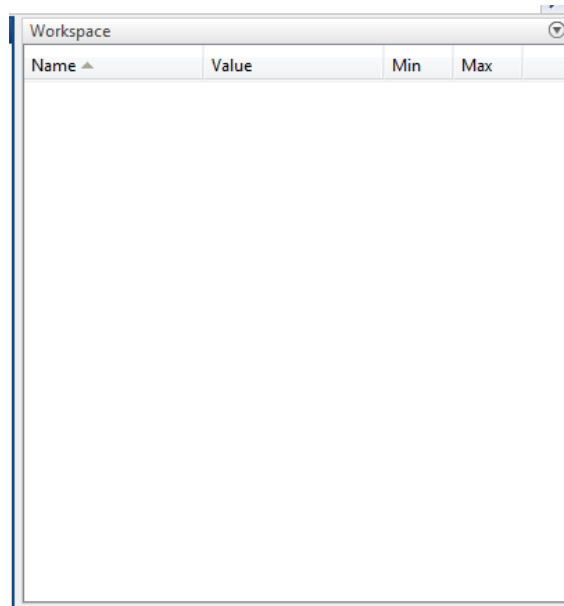
Para abrir el editor sin abrir un archivo ya existente:
Esquina superior izquierda, en “New”, abrir script.....
el editor se abre con un archivo sin título al que vosotros le tenéis que poner nombre cuando lo salvéis.



•Command window

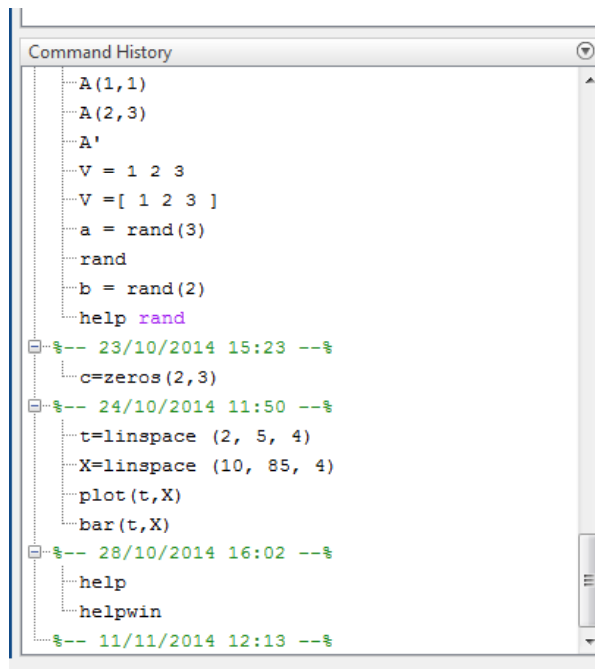
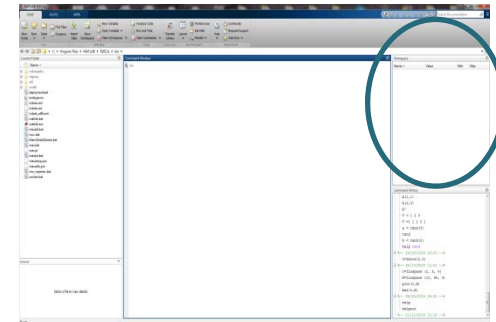
Donde se ejecutan los comandos de MATLAB: después del >> (prompt) que indica que el programa está preparado para recibir instrucciones. **Es también donde se muestran los resultados.** Es la única ventana de **ejecución**, las demás son informativas.

Se permiten líneas de comandos muy largas que automáticamente siguen en la línea siguiente al llegar al límite del margen derecho de la ventana (se activa la opción Wrap Lines, en el menú File/Preferences/Command Window).



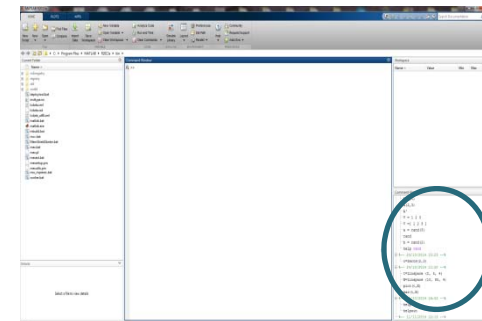
•Work space

Contiene **información** sobre todas las variables que se hayan definido en la sesión y permite ver y modificar las matrices con las que se está trabajando. Para obtener información de las variables desde la línea de comandos se pueden utilizar el comando *whos*. *whos* da información de las variables que se manejan en ese momento.



•Command history

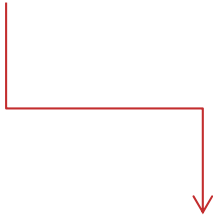
Muestra los **últimos comandos ejecutados** en la **command window**. Se pueden volver a ejecutar haciendo doble click sobre ellos. Son también accesibles con la teclas \uparrow y \downarrow .



VARIABLE CODE SIMULINK ENVIRONMENT

matica aplicada pen CARPETA PRINCIPAL ▶ prog ▶ Programas ▶

```
Command Window  
>> y  
Undefined function or variable 'y'.  
f1 >>
```



```
Command History  
-- 27/11/2015 12:01 --  
└─ y
```



```
Command Window
>> y
Undefined function or variable 'y'.

>> Y=2

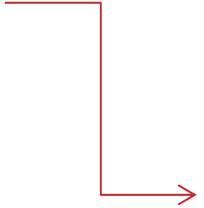
Y =

     2

fx >> |
```

Workspace

Name	Value	Min	Max
Y	2	2	2



```
Command History
-- 27/11/2015 12:01 --
├─ y
└─ Y=2
```

```
Command Window
>> y
Undefined function or variable 'y'.

>> Y=2

Y =

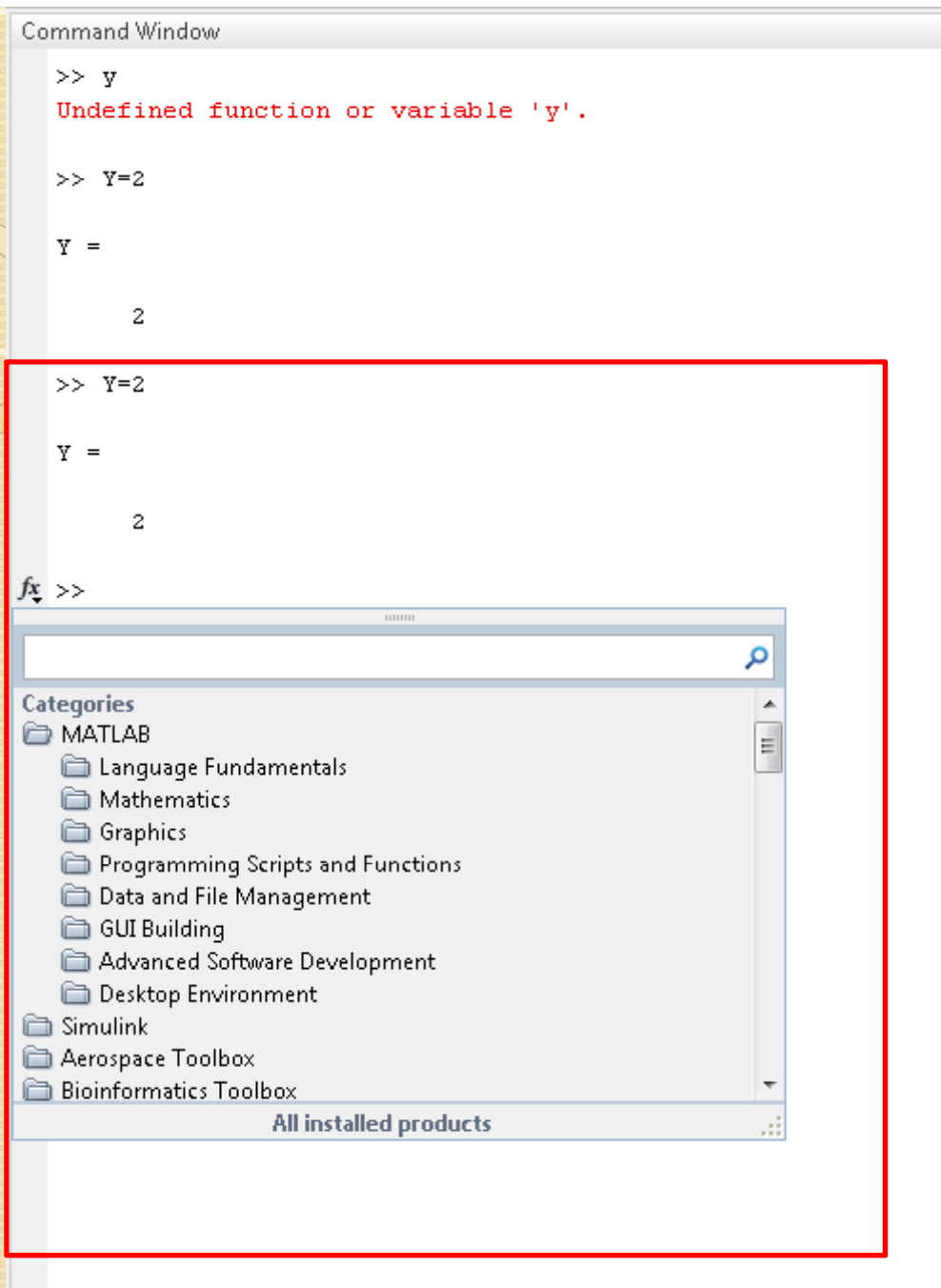
    2

>> Y=2

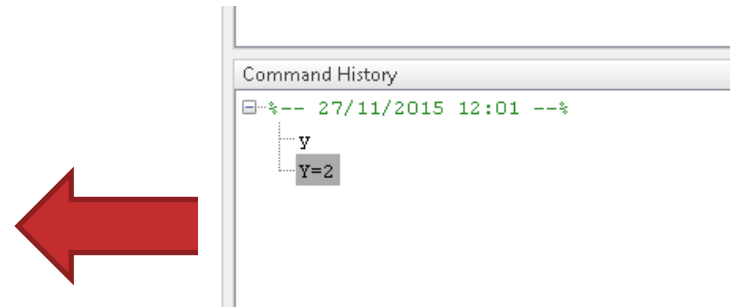
Y =

    2

fx >>
```



```
Command History
---%-- 27/11/2015 12:01 ---%
  y
  Y=2
```



-En Command window clicando sobre un comando con el botón derecho se muestra un menú contextual con las posibilidades disponibles en ese momento. Para editar uno de estos comandos hay que copiarlo antes al **command window** (o clicando 2 veces botón iz.)

1.1 Normas iniciales de uso

-Para borrar todas las salidas de MATLAB y dejar limpia la Command window se puede utilizar las funciones `clc` y `home`:

“La función `clc` (clear console) elimina todas las salidas anteriores y `home` las mantiene, pero lleva el prompt a la primera línea de la ventana”.

También se puede utilizar `clear all`; `clear functions`: sólo elimina las funciones; `clear name*` elimina las variables que empiezan por `name`.

Además cada ventana tiene un dialogo donde borrarlo todo.

Si se quiere salir de MATLAB se teclea los comandos

`quit` o `exit`, elegir `exit` en MATLAB.

AYUDA EN MATLAB

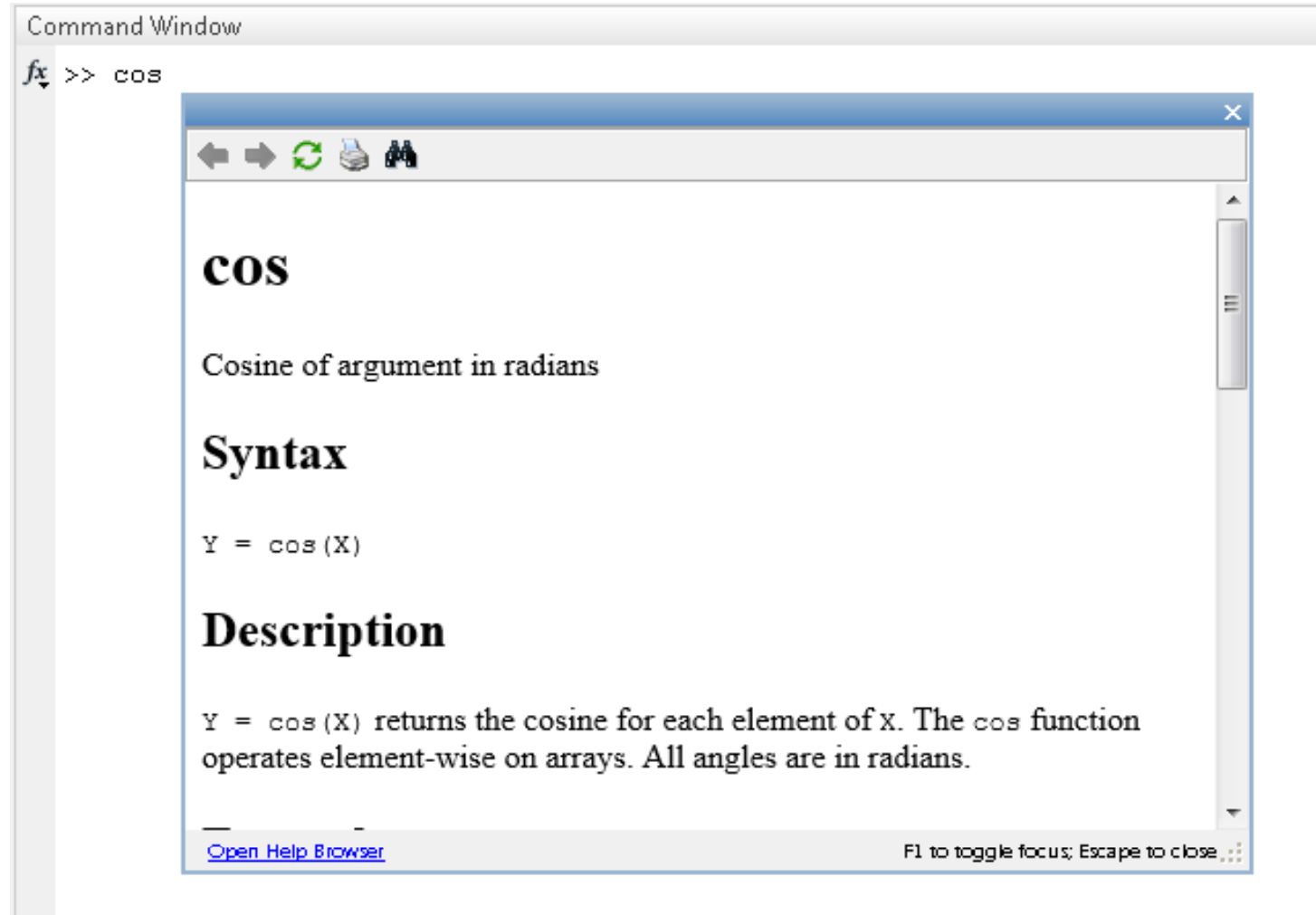
Command Window


```
>> helpwin cos  
fx >>
```



-El comando helpwin seguido de un nombre de comando o de función muestra la información correspondiente a ese comando en la ventana Help.

-Help desde la línea de comandos de la Command Window, clicando con el botón derecho sobre el nombre de la función. Es recomendable usarla para obtener una información más precisa sobre la sintaxis y diversas posibilidades de uso de los comandos.





MATLAB contiene muchas funciones matemáticas predeterminadas, visibles con 'help elfun'. Las funciones especiales se obtienen con 'help specfun'.

Elementary math functions.

Trigonometric.

sin - Sine.
sind - Sine of argument in degrees.
sinh - Hyperbolic sine.
asin - Inverse sine.
asind - Inverse sine, result in degrees.
asinh - Inverse hyperbolic sine.
cos - Cosine.
cosd - Cosine of argument in degrees.
cosh - Hyperbolic cosine.
acos - Inverse cosine.
acosd - Inverse cosine, result in degrees.
acosh - Inverse hyperbolic cosine.
tan - Tangent.
tand - Tangent of argument in degrees.
.....

Exponential.

Complex.

Rounding and remainder.

erf - Error function

>> help erf

This MATLAB function returns the Error Function evaluated for each element of x.

erf(x)

See also erfc, erfcinv, erfcx, erfinv

Reference page for erf

Other functions named erf

'lookfor' permite buscar los comandos o ficheros de MATLAB relacionados con una palabra clave que nos interese. Una vez encontrados, el comando 'help' nos dará información específica sobre los comandos que nos interesen.

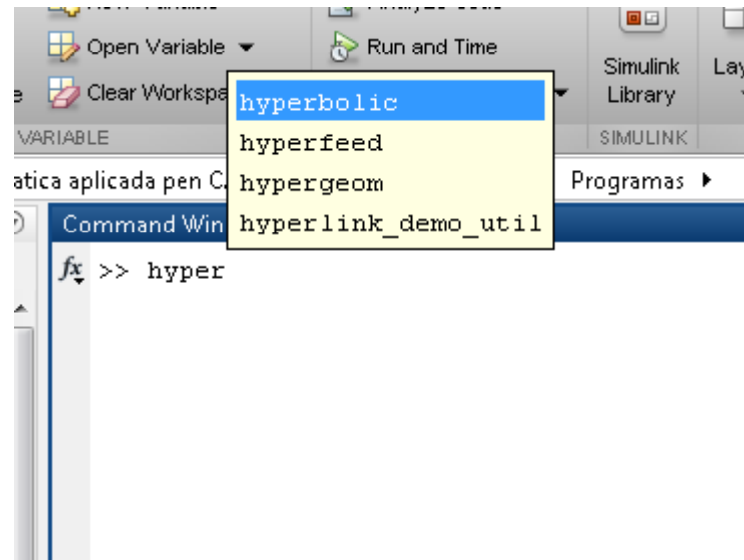
>> lookfor 'differential equation' (las comillas delimitan una cadena de caracteres)

>> help ode15s

1.1 Normas iniciales de uso.

Problema: no sé con certeza el nombre de una función

-Comenzando a teclear el nombre de la función y pulsando la tecla Tab(tabulador), MATLAB completa automáticamente el nombre de la función, o bien muestra en la línea siguiente todas las funciones disponibles que comienzan con esas teclas.



Recurso: *Con el cursor hacia arriba, recuperamos las sentencias anteriores y cambiamos las comas por puntos y comas, o lo que necesitemos cambiar*

-COMETEMOS UN ERROR EN MATLAB: Cuando al ejecutar un fichero *.m o una operación determinada, puede producirse un error.



-La respuesta de MATLAB es un mensaje en el Command Window que muestra el error en color rojo.

-Si es un archivo, mediante un subrayado el enlace a la línea del fichero fuente en la que se ha producido el error (**dice donde está el error**).

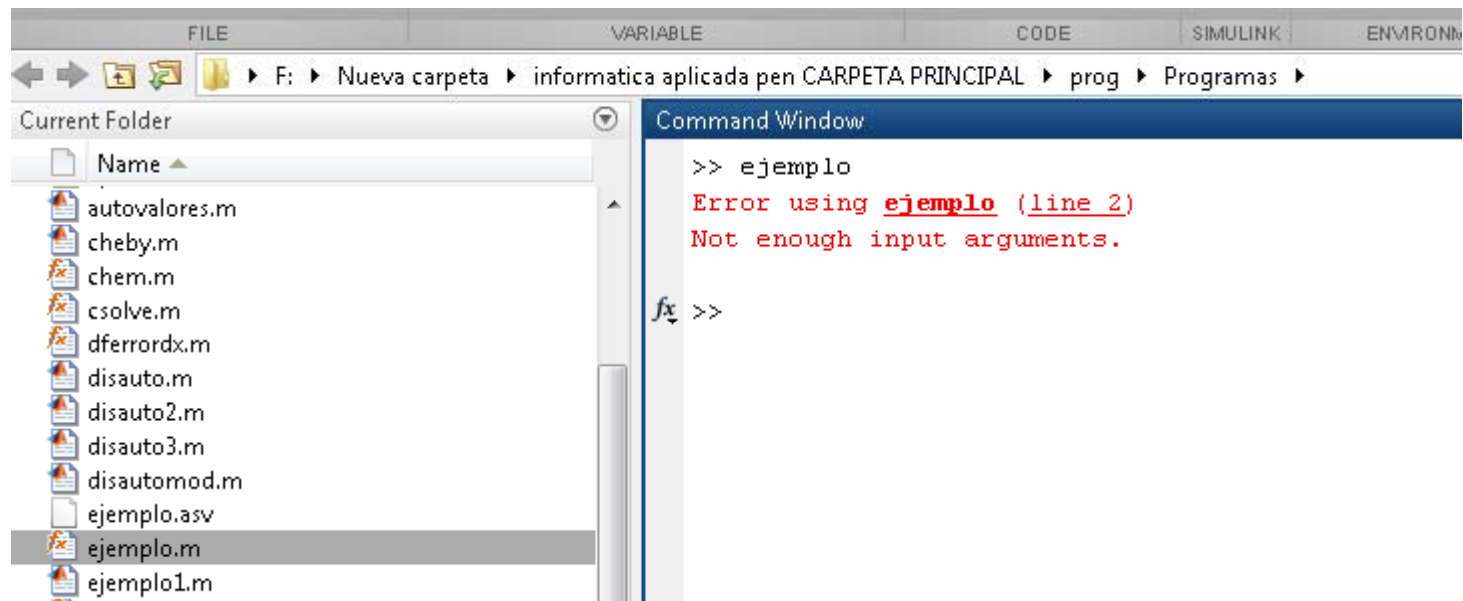
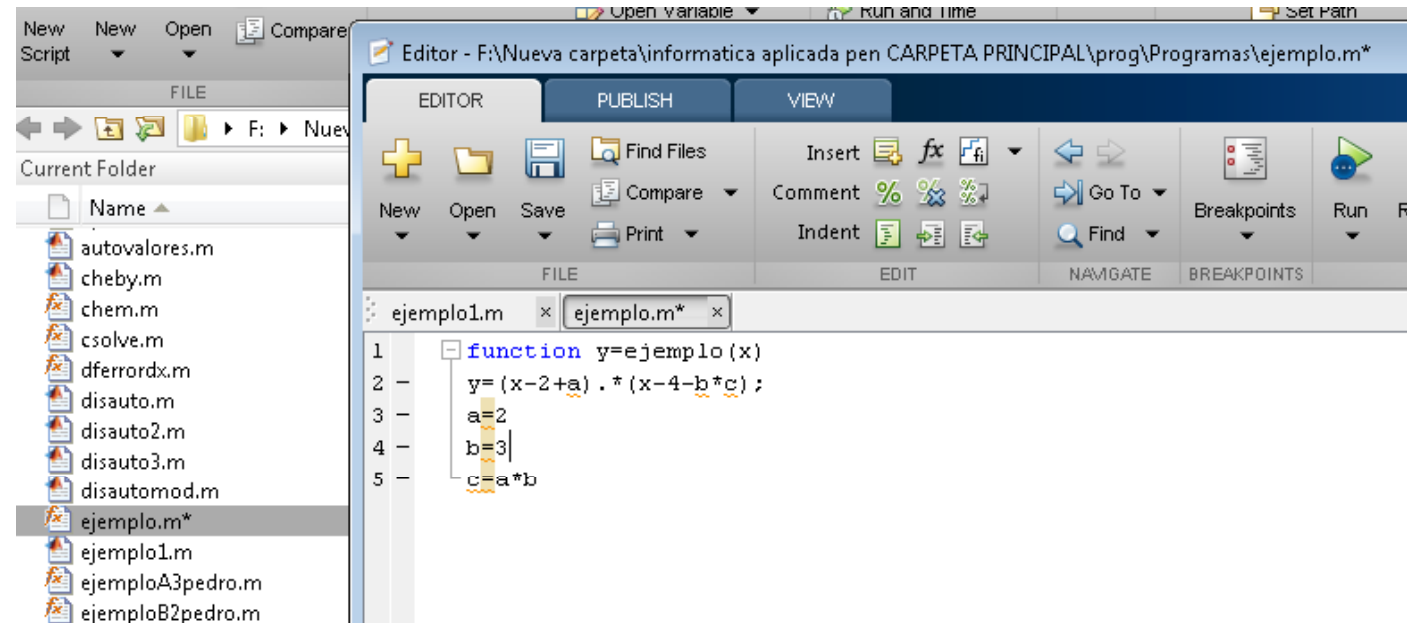
Clicando en ese enlace se va a la línea correspondiente del fichero por medio de Editor/(Debugger).

Ejemplo:

A screenshot of the MATLAB Command Window. The window title is "Command Window". The command prompt shows the input ">> 2 * y". Below the input, the error message "Undefined function or variable 'y'." is displayed in red text. The window's path bar shows "matlab aplicada pen CARPETA PRINCIPAL > prog > Programas >".

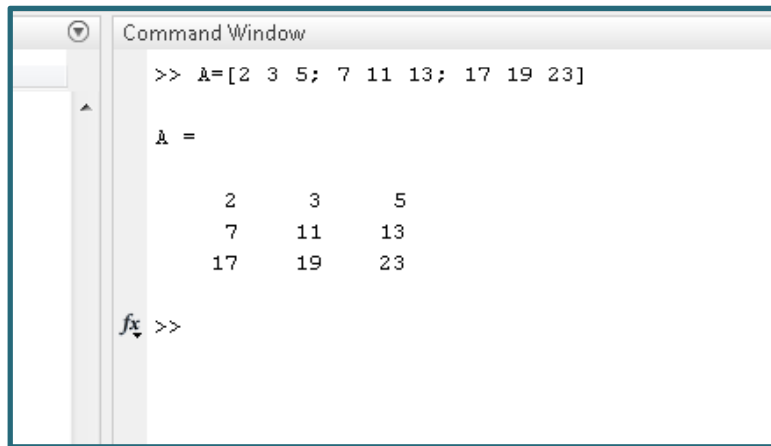
```
matlab aplicada pen CARPETA PRINCIPAL > prog > Programas >  
Command Window  
>> 2 * y  
Undefined function or variable 'y'.  
fx >>
```

Ejemplo:



1.1 Normas iniciales de uso.

-Array Editor (**Workspace**): editor de vectores y matrices. No sólo permite ver los valores de los elementos de cualquier matriz o vector definido en el programa, es también posible *modificar estos valores* clicando sobre la celda correspondiente. La ventana del *Array Editor* incluye una lista desplegable en la que se puede elegir el formato en el que se desea ver los datos.

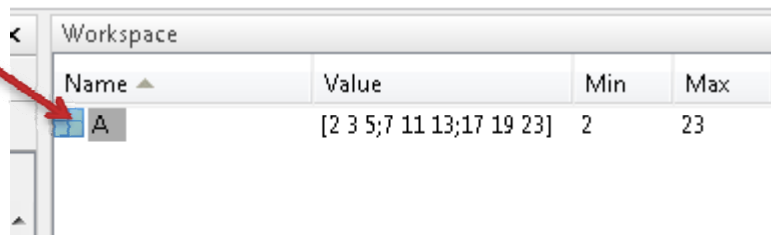


```
Command Window
>> A=[2 3 5; 7 11 13; 17 19 23]

A =

     2     3     5
     7    11    13
    17    19    23

fx >>
```



Name	Value	Min	Max
A	[2 3 5; 7 11 13; 17 19 23]	2	23



Variables - A

A x

A <3x3 double>

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

Command Window

```
>> A=[2 3 5; 7 11 13; 17 19 23]

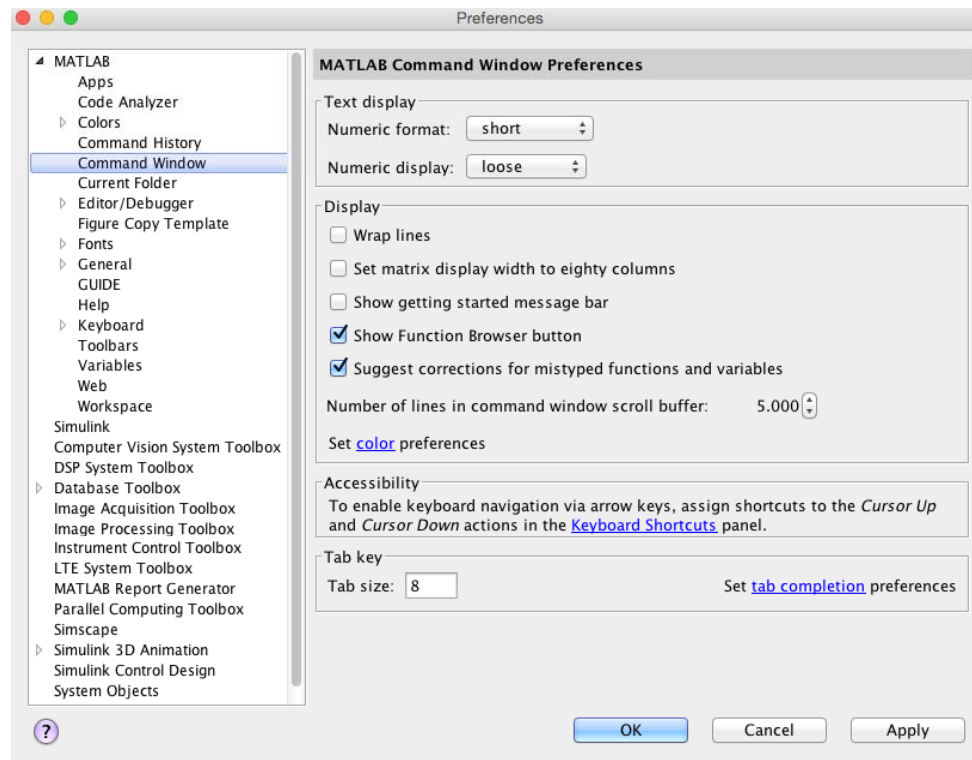
A =

     2     3     5
     7    11    13
    17    19    23

fx >>
```

1.1 Normas iniciales de uso.

-**Formatos de salida y de otras opciones de MATLAB:** dispone de un cuadro de diálogo que permite distintas opciones de presentación (más de 20 cuadros de diálogo); se abre con *Preferences* del menú *File*.



Posibilidades de elegir color, tipo de letra. Tamaño, etc. (*Command Window/Fonts*) (procurar elegir siempre del tipo *Courier New*, *Lucida Console* o *Monospaced*).

1.1 Normas iniciales de uso.

- **Formatos numéricos** con que MATLAB muestra los resultados (siempre calcula con doble precisión 16 cifras decimales equivalentes)

Posibilidades:

- short fija con 4 decimales (por defecto)
- long fija 15 decimales
- hex cifras hexadecimales
- bank números con dos cifras decimales
- short e notación científica con 4 decimales
- short g notación científica o decimal
- long e notación científica con 15 decimales
- long g notación científica o decimal.

Estos formatos se pueden cambiar también desde la línea de comandos anteponiendo la palabra *format*. Por ejemplo, para ver las matrices en formato *long* habrá que ejecutar el comando:

```
>> format long
```

1.1 Normas iniciales de uso.

Ejemplo de cómo cambiar el formato de una matriz hecha:

```
Command Window
>> a = [0.1 0.002 0.0035; 0.0007 0.003 0.000897]

a =

    0.1000    0.0020    0.0035
    0.0007    0.0030    0.0009

>> format long
>> a

a =

    0.100000000000000000    0.002000000000000000    0.003500000000000000
    0.000700000000000000    0.003000000000000000    0.000897000000000000

fx >>
```

1.1 Normas iniciales de uso.

```
>>format short  
>> sqrt(2)  
ans = 1.4142
```

```
>>format long  
>> sqrt(2)  
ans= 1.41421356237310
```

```
>>format long e  
>> sqrt(2)  
ans= 1.41421356237310 e*00
```

```
>>format short  
>>A=[1000 0.0001]  
ans=1.0 e*03 =1.0000 0.0000
```

```
>>format rat  
>>A  
A =  
10000 1/10000
```

1.1 Normas iniciales de uso.

-El formato *loose* introduce algunas líneas en blanco en la salida (opción por defecto), mientras que el formato *compact* elimina las líneas en blanco citadas.

- Guardar variables y estados de una sesión: Comandos *save* y *load*

>> **save**, se crea en el *directorio actual* un fichero binario llamado *matlab.mat* (o *matlab*) con el estado de la sesión (excepto los gráficos).

Dicho estado puede recuperarse la siguiente vez que se arranque el programa con el comando >> **load**

Se pueden guardar también matrices y vectores de forma selectiva y en ficheros con nombre especificado por el usuario. Por ejemplo, el comando (sin comas entre los nombres de variables): >> **save filename A x y** guarda las variables **A**, **x** e **y** en un fichero binario llamado *filename.mat* (o *filename*). Para recuperarlas en otra sesión basta teclear >> **load filename**

El comando *save* permite guardar el estado de la sesión en formato ASCII.

1.1 Normas iniciales de uso.

- Guardar información de la sesión y copiar salidas: Comando *diary*

Existe una forma sencilla de almacenar en un fichero un texto que describa lo que el programa va haciendo (la entrada y salida de los comandos utilizados). Esto se hace con el comando *diary* en la forma siguiente:

>> **diary filename.txt**

>> **diary off** (suspende la ejecución de *diary*)

>> **diary on** (*diary on* la reanuda)

Para poder acceder al fichero *filename.txt* con *Notepad* o *Word* es necesario que *diary* esté en *off*.

- Líneas de comentarios

El carácter *tanto por ciento* (%) indica comienzo de comentario. Cuando aparece en una línea de comandos, el programa supone que todo lo que va desde ese carácter hasta el fin de la línea es un comentario. MATLAB ignora todo lo que vaya precedido por el símbolo %

1.2 Variables, símbolos, operadores y funciones de MATLAB:

Los **nombres de las variables** deben seguir estas reglas:

- 1.- Se forman con las letras del abecedario, los dígitos de 0 a 9 y el signo “_”
- 2.- Los nombres de las variables han de comenzar por una letra y no deben contener espacios en blanco.
- 3.- Sólo se reconocen los 31 primeros caracteres de nombre, el resto no aparece.
- 4.- Los nombres de las variables no pueden coincidir con los nombres de las keywords (nombres reservados). La lista de los nombres reservados se obtiene por medio de **lists iskeyword (en ayuda)**: aparece un enlace.
- 5.- **MATLAB** distingue entre mayúsculas y minúsculas.

6.- Los cálculos que no se asignan a una variable en concreto se asignan a la variable de respuesta por defecto que es ans (del inglés, *answer*); si el cálculo se asigna a una variable, el resultado queda en esa variable:

```
>>2+3
ans =
5
>>x=2+3
x =
5
```

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

```
>>x
x =
5
```

- **Si se introducen los números con signo, no dejar un espacio entre el signo y el número**

1.2 Variables, símbolos, operadores y funciones de MATLAB:

8.- La comilla ' es la que, en un teclado estándar, se encuentra en la tecla de la interrogación

9.- En MATLAB, las comas y los puntos y comas se usan, entre otras cosas, para separar sentencias que se han escrito en la misma línea:

Si se añade un punto y coma (;) al final de la instrucción, la máquina no muestra la respuesta pero no por ello deja de realizarse el cálculo. Con comas sin embargo aparecen los valores de todas las variables que debe calcular.

¿Qué diferencia hay entre estas dos expresiones?

```
>> x=2; y=cos(0.3); z=3*x*y
```

```
>> x=2, y=cos(0.3), z=3*x*y
```

```
Command Window
>> x=2; y=cos(.3);z=3*x*y

z =

    5.732018934753636

>> x=2, y=cos(.3),z=3*x*y

x =

    2

y =

    0.955336489125606

z =

    5.732018934753636

fx >> |
```

10.- Los operadores aritméticos son:

+,- **suma, resta**

***, /** **multiplicación, división**

^ **elevado a**

11.- Para operadores de igual prioridad, se ejecuta primero el que está a la izquierda.

12.- Se pueden emplear paréntesis para anular el orden de prioridad (ejecutándose siempre en primer lugar lo que está dentro de los paréntesis más internos).

Algunos ejemplos:

>> 2+3*4

ans = 14

>> 1+2/3*4

ans = 3.667

>> -2-3*4

ans =-14

>> 1+2/(3*4)

ans = 1.1667

1.2 Variables, símbolos, operadores y funciones de MATLAB:

13.- El logaritmo neperiano se escribe 'log' y el decimal 'log10'. Se pueden utilizar las funciones matemáticas habituales. Así, por ejemplo, la función coseno,

```
>>cos(pi) % pi es una variable con valor predeterminado 3.14159...
```

```
ans =
```

```
-1
```

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

```
ans =
```

```
2.7183
```

14.- Además de la variable pi, MATLAB tiene otras variables con valor predeterminado; éste se pierde si se les asigna otro valor distinto.

```
>>eps % Épsilon de la máquina.
```

```
ans=2.2204e-016
```

```
pero...
```

```
>>eps=7
```

```
eps = 7
```

1.2 Variables, símbolos, operadores y funciones de MATLAB:

15.- Trabajar con complejos no da ningún tipo de problema. La unidad imaginaria se representa en **MATLAB** como i o j , variables con dicho valor como predeterminado:

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

```
ans =  
0+ 2.0000i
```

Ejemplo: Definimos una matriz compleja como

```
C=[3+4i ; 5+4j]
```

```
C =
```

```
3.0000 + 4.0000i
```

```
5.0000 + 4.0000i
```

Su parte real e imaginaria puede extraerse con los comandos `real` e `imag`. El valor absoluto puede recuperarse con `abs` y el ángulo en radianes con `angle`.

1.2 Variables, símbolos, operadores y funciones de MATLAB:

$C=[3+4i ; 5+4j];$

Parte real de matriz C:	<code>real (C)</code>	$C_r = 3$ 5
Parte imaginaria de matriz C:	<code>imag (C)</code>	$C_i = 4$ 4
<code>abs(C)</code> : valor absoluto de los elementos de la matriz.		$=5.0000$ 6.4031

1.3 Matrices en MATLAB:

- Una matriz de $m \times n$ elementos es una disposición bidimensional de números con m filas (horizontales) y n columnas (verticales)

¿Cómo escribir matrices y vectores?

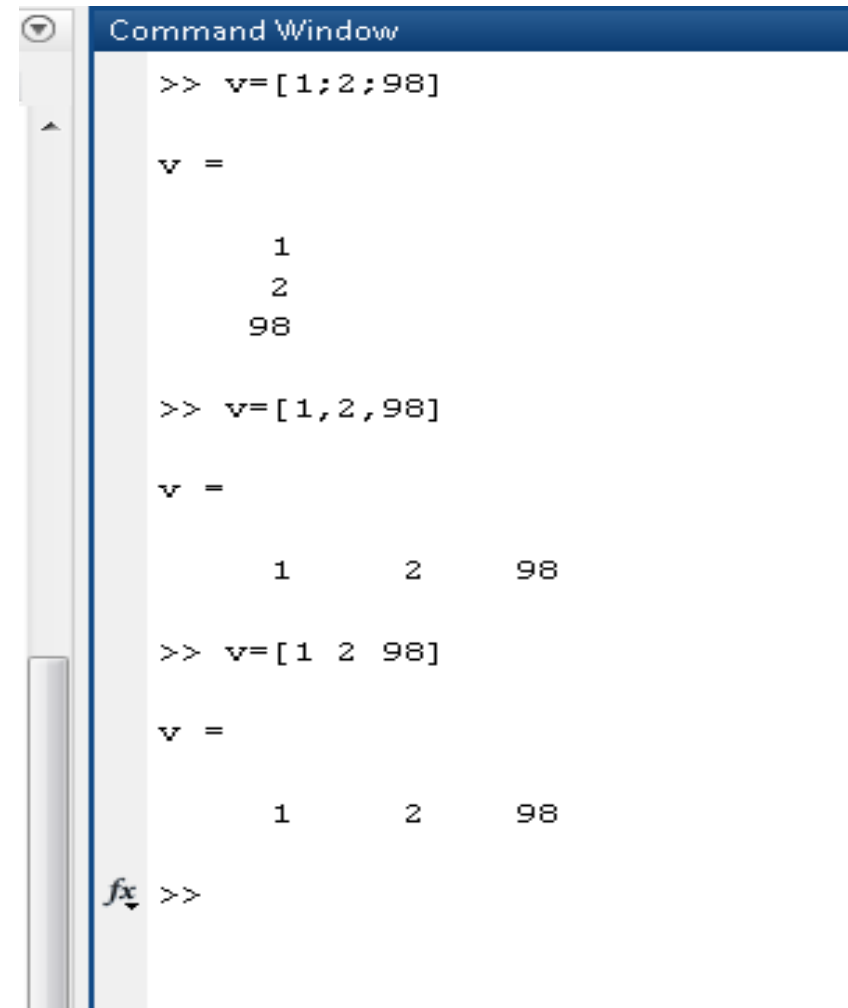
COMA, ESPACIO y PUNTO Y COMA

Una matriz puede definirse usando corchetes, **comas o espacios**, y **puntos y comas**

```
>> A=[ 2 3 5 ; 7 11 13 ; 17 19 23 ]
```

ó

```
>> A=[ 2,3,5 ;7,11,13 ;17,19,23 ]
```



```
Command Window
>> v=[1;2;98]
v =
     1
     2
    98
>> v=[1,2,98]
v =
     1     2    98
>> v=[1 2 98]
v =
     1     2    98
fx >>
```

1.3 Matrices en MATLAB:

- Para definir un vector fila, basta introducir sus coordenadas entre corchetes:

```
>>v=[1 8 9] % Vector de 3 coordenadas
```

```
v=
```

```
1 8 9
```

- El carácter ‘:’ se usa para definir vectores:

```
>> 1:5
```

```
1 2 3 4 5
```

- Para espaciados distintos de uno, el espaciado s se introduce i:s:j

```
>> 0:0.1:1
```

```
ans=
```

```
0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000
```

1.3 Matrices en MATLAB:

- Algunas matrices elementales se generan con funciones de MATLAB:

`>> eye(2)`  `>>eye(5) % eye diagonal 1` matriz identidad,

`>> zeros(2)` 

`>> ones(2)` 

la matriz nula,

`>>zeros(3)`

ans =

0 0 0

0 0 0

0 0 0

ans =

1 0 0 0 0

0 1 0 0 0

0 0 1 0 0

0 0 0 1 0

0 0 0 0 1

o la matriz cuyos elementos valen todos 1:

`>>ones(4)`

ans =

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

1.3 Matrices en MATLAB:

Se puede conocer el tamaño de una matriz y la longitud de un vector:

```
mat=[2 3 1; 4 5 3; 6 3 8]
```

```
>>size(mat) % Dimensiones de la matriz mat (número de filas y de columnas)
```

```
ans =
```

```
3 3
```

```
>>V= [1 2 3]
```

```
>>size(V)
```

```
ans =
```

```
1 3
```

```
>>length(V) % Longitud del vector (número de coordenadas)
```

```
ans =
```

```
3
```

- Otras funciones generadoras de matrices elementales pueden obtenerse con 'help elmat'

1.3 Matrices en MATLAB:

```
(A era A=[ 2 3 5 ; 7 11 13 ; 17 19 23 ])
```

2	3	5
7	11	13
17	19	23

- Para hacer la traspuesta:

```
>>A' % Su traspuesta (su adjunta)
```

El operador ' es el de trasposición (en realidad trasposición y conjugación):

```
>>A'
```

```
ans =
```

2	7	17
3	11	19
5	13	23

Existen comandos que permiten crear de forma sencilla matrices. Siendo el vector
 $v = (1 \ 2 \ 3)$

```
>>diag(v) % Matriz diagonal cuya diagonal es el vector v
```

```
ans =
```

1	0	0
0	2	0
0	0	3

1.3 Matrices en MATLAB:

>>diag(diag(A)) % Matriz diagonal con la diagonal de A. La sentencia
diag(A) da el vector formado por la diagonal de la matriz A

ans =

```
2 0 0
0 11 0
0 0 23
```

2	3	5
7	11	13
17	19	23

>>diag(ones(1,4),1)+diag(ones(1,4),-1) % Matriz tridiagonal 5x5 con 0
en la diagonal principal y 1 en la sub y superdiagonal

ans =

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

1.3 Matrices en MATLAB:

```
>>tril(A) % Matriz formada por la parte triangular inferior de A.
```

```
ans =
```

```
2 0 0
7 11 0
17 19 23
```

2	3	5
7	11	13
17	19	23

```
>>triu(A) % Matriz formada por la parte triangular superior de A.
```

```
ans =
```

```
2 3 5
0 11 13
0 0 23
```

- Los elementos de una matriz se extraen escribiendo el nombre de la matriz y, entre paréntesis, los subíndices correspondientes a la posición.

```
>>A(1,3) % Elemento en la primera fila y tercera columna de la matriz A
```

```
ans =
```

```
5
```

1.3 Matrices en MATLAB:

- Dos puntos indica “todas las filas” o “todas las columnas”:

```
>> a=rand(3)
```

```
>> a(:,2)
```

```
>> a(2,:)
```

rand: crea un número, vector o matriz aleatorio

```
Command Window
>> a=rand(3)

a =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575

>> a(:,2)

ans =

    0.9134
    0.6324
    0.0975

>> a(2,:)

ans =

    0.9058    0.6324    0.5469

fx >>
```


Más ejemplos del comando rand

Command Window

```
>> rand
```

```
ans =
```

```
0.9649
```

```
>> rand(1,3)
```

```
ans =
```

```
0.1576    0.9706    0.9572
```

```
fx >> |
```

Función redondeo: round

Command Window

```
>> round (2.345678)
```

```
ans =
```

```
2
```

```
fx >>
```

1.3 Matrices en MATLAB:

- Función 'linspace'. ¿Qué es? Usamos la ayuda.

```
>> help linspace
```

- El número de incrementos que genera entre x1 y x2 es N-1 (separación=(x2-x1)/(N-1))

- Ejemplo:

```
>> linspace (0,0.5,11)
```

```
0  0.0500  0.1000  0.1500  0.2000  0.2500  0.3000  0.3500  0.4000  0.4500  0.5000
```

Se consigue lo mismo con: 0:0.05:0.5

```
0  0.0500  0.1000  0.1500  0.2000  0.2500  0.3000  0.3500  0.4000  0.4500  0.5000
```

Ejemplo:

```
>> linspace (0,0.5,30)
```

```
ans =
```

```
Columns 1 through 14
```

```
    0    0.0172    0.0345    0.0517    0.0690  
0.0862    0.1034    0.1207    0.1379    0.1552  
0.1724    0.1897    0.2069    0.2241
```

```
Columns 15 through 28
```

```
    0.2414    0.2586    0.2759    0.2931    0.3103  
0.3276    0.3448    0.3621    0.3793    0.3966  
0.4138    0.4310    0.4483    0.4655
```

```
Columns 29 through 30
```

```
    0.4828    0.5000
```

Otro ejemplo de linspace: `>>linspace(1,11)`

No especificamos número de intervalos (N-1). Por defecto 100

```
Command Window
>> linspace(1,11)

ans =

Columns 1 through 13
    1.0000    1.1010    1.2020    1.3030    1.4040    1.5051    1.6061    1.7071    1.8081    1.9091    2.0101    2.1111    2.2121

Columns 14 through 26
    2.3131    2.4141    2.5152    2.6162    2.7172    2.8182    2.9192    3.0202    3.1212    3.2222    3.3232    3.4242    3.5253

Columns 27 through 39
    3.6263    3.7273    3.8283    3.9293    4.0303    4.1313    4.2323    4.3333    4.4343    4.5354    4.6364    4.7374    4.8384

Columns 40 through 52
    4.9394    5.0404    5.1414    5.2424    5.3434    5.4444    5.5455    5.6465    5.7475    5.8485    5.9495    6.0505    6.1515

Columns 53 through 65
    6.2525    6.3535    6.4545    6.5556    6.6566    6.7576    6.8586    6.9596    7.0606    7.1616    7.2626    7.3636    7.4646

Columns 66 through 78
    7.5657    7.6667    7.7677    7.8687    7.9697    8.0707    8.1717    8.2727    8.3737    8.4747    8.5758    8.6768    8.7778

Columns 79 through 91
    8.8788    8.9798    9.0808    9.1818    9.2828    9.3838    9.4848    9.5859    9.6869    9.7879    9.8889    9.9899    10.0909

Columns 92 through 100
    10.1919    10.2929    10.3939    10.4949    10.5960    10.6970    10.7980    10.8990    11.0000
```

1.4 Operadores para matrices

- Los operadores +,-,*,/ operan con matrices siguiendo las reglas del álgebra matricial
- Existe también el operador '\'. 'b\a' equivale a 'a/b'
- Los operadores elemento a elemento operan con los elementos con igual posición en las matrices implicadas (SÓLO CON MATRICES CON DIMENSIONES IGUALES).

$$D = [1 \ 2 \ 3] + [2 \ 4 \ 6]$$

$$D = \begin{matrix} 3 & 6 & 9 \end{matrix}$$

$$D = [1 \ 2 \ 3] * [2; 4; 6]$$

$$D = 28$$

1.4 Operadores para matrices

Cuando uno de los operadores es una matriz y el otro es un escalar:

$$D=[1 \ 2 \ 3]-2$$

$$D=3*[1 \ 2 \ 3]-2$$

$$D= \begin{matrix} -1 & 0 & 1 \end{matrix}$$

$$D= \begin{matrix} 1 & 4 & 7 \end{matrix}$$

Exponencial:

$$D=[1 \ 2; 3 \ 4]^3$$

$$D = \begin{matrix} 37 & 54 \\ 81 & 118 \end{matrix}$$

2 operadores para “división de matrices):

Símbolo “\” división izquierda (**mldivide**).
Si $X = A \setminus B = \text{inv}(A) * B$; $A * X = B$. Matrices con las mismas o distintas dimensiones.

Símbolo “/” división derecha (**mrdivide**).
Matrices con las mismas dimensiones. No funciona si las matrices tienen distintas dimensiones.

Command Window

```
>> D=[1;2]\[1 2;3 4]
```

```
D =
```

```
    1.4000    2.0000
```

```
>> D=[1 2;3 4]\[1;2]
```

```
D =
```

```
    0  
    0.5000
```

```
>> D=[1;2]/[1 2;3 4]
```

```
Error using /  
Matrix dimensions must agree.
```

```
>> D=[1 2;3 4]/[1;2]
```

```
Error using /  
Matrix dimensions must agree.
```

```
>> D=[1 4 ; 2 6]/[1 2 ; 3 4]
```

```
D =
```

```
    4.0000   -1.0000  
    5.0000   -1.0000
```

```
fx >> |
```

1.4 Operadores para matrices

Si se desea es realizar una operación determinada sobre cada elemento de una matriz, se antepone al correspondiente operador el símbolo de punto (.) para que Matlab realice la correspondiente operación elemento a elemento.

```
>> a.*b
```

```
D=[1 2 3].*[1 2 3]    D=
                       1   4   9
```

```
D=[1 2 3].^[1 2 3]   D=
                       1   4  27
```

Ejemplo: Hacer

```
>> a=rand(3)
```

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

```
>> a*b
```

```
>> exp a
```

```
>> sqrt a
```


1.5 Operadores comparativos y lógicos. Control del flujo

- Los operadores comparativos de MATLAB son:

== igual a

> mayor que

~= no igual a

<= menor o igual que

< menor que

>= mayor o igual que

- Los operadores lógicos son:

& y lógico (logical and)

| o lógico (logical or)

~ no lógico

xor o exclusivo

Símbolos como & ~ |

1.5 Operadores comparativos y lógicos.

- Es conveniente utilizar paréntesis para ejecutar las comparaciones correctamente

Ejemplos: $(a > b) \& (b \neq 0)$

$(a > b) | (b \neq 0)$

También hay variables lógicas que toman los valores 0 (falso) o 1 (verdadero) . Por ejemplo:

$v = 1 \ 2 \ 3$

```
>>abs(v) >= 2      % Vector lógico cuyas coordenadas valen 1 si la coordenada  
es >= 2 y 0 si no lo es
```

```
ans=  
0 1 1
```

```
>>vector=v(abs(v)>=2) % Vector formado por la coordenadas de v que  
% verifican que sea mayor o igual a 2.
```

```
vector =  
2 3
```

1.5 Operadores comparativos y lógicos.

$v = [1 \ 2 \ 3]$ $v2 = [3 \ 2 \ 1]$

```
>>logica=v==v2 % Asignación de un valor lógico (el  
doble signo igual es el  
% igual lógico)
```

```
logica =  
0 1 0
```

```
>>logic2=v~=v2 % Distinto (~ es el operador de  
negación)
```

```
logic2 =  
1 0 1
```