

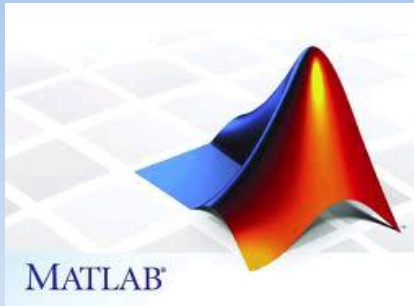
# Unidad 1

## **INTRODUCCIÓN A MATLAB**

# Contenido

- Introducción a Matlab
- Herramientas del escritorio
- Variables, definición y generación.
- Concatenación e indexación.
- Operadores aritméticos.
- Funciones matemáticas básicas.
- Funciones sobre matrices y vectores.
- Funciones gráficas.

# Introducción



**Matlab** es un paquete software altamente especializado para realizar cálculos científicos y técnicos, especialmente los relacionados con matrices y vectores.

Además de los cálculos una de las capacidades más atractivas son las herramientas de presentación gráfica, tanto en 2D como en 3D.

Matlab cuenta con un lenguaje de programación para interactuar con el usuario, siendo dicho lenguaje independiente de la plataforma en la que esté corriendo el programa.

# Introducción

## Competencia

Maple



Ventajas: Software libre.

Mathematica



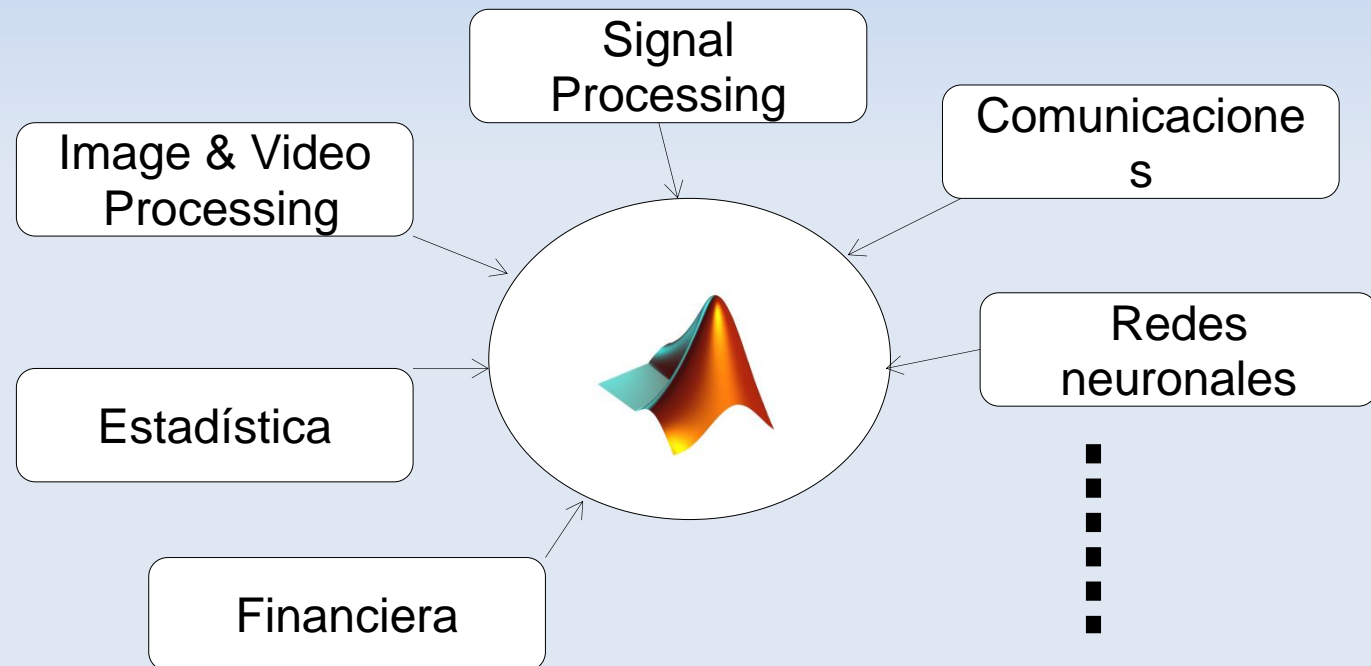
Desventajas: No están tan extendidos como Matlab

R



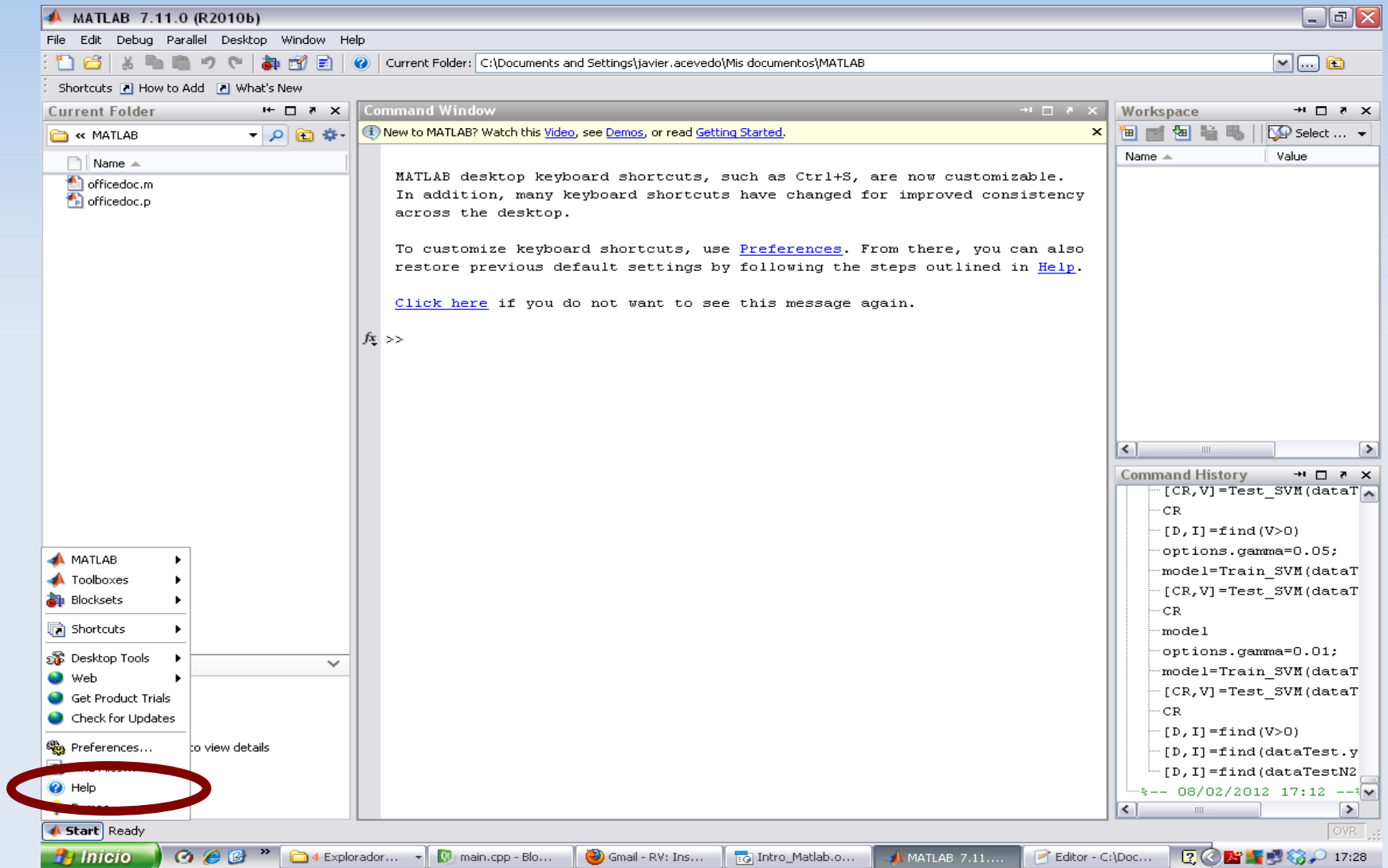
# Introducción

**Toolbox:** Conjunto de funciones que forman una librería.  
Son desarrollos focalizados en diferentes campos y de los cuales se ha desarrollado código específico que facilita trabajar en ese campo.



Mathworks comercializa algunas de estas toolbox, pero en otros casos son empresas externas o usuarios las que las desarrollan.

# Ayuda de Matlab



# Ayuda de Matlab

The screenshot shows the MATLAB Help application window. The title bar reads 'Help'. The menu bar includes 'File', 'Edit', 'View', 'Go', 'Favorites', 'Desktop', 'Window', and 'Help'. A search bar is located at the top left. On the left side, there is a 'Contents' pane with a tree view. The 'MATLAB' folder is expanded, and 'Getting Started' is selected. Below it, a list of topics is shown, including 'Introduction', 'Matrices and Arrays', 'Graphics', 'Programming', 'Data Analysis', 'Creating Graphical User Interfaces', 'Desktop Tools and Development Environment', 'External Interfaces', 'User's Guide', 'Functions', 'Examples', 'Demos', and various toolboxes like 'Control System Toolbox', 'Curve Fitting Toolbox', 'Database Toolbox', 'Datafeed Toolbox', 'Global Optimization Toolbox', 'Image Acquisition Toolbox', 'Image Processing Toolbox', 'MATLAB Builder EX', 'MATLAB Builder JA', 'MATLAB Builder NE', 'MATLAB Compiler', 'MATLAB Distributed Computing Server', 'MATLAB Report Generator', 'Neural Network Toolbox', 'OPC Toolbox', 'Optimization Toolbox', 'Parallel Computing Toolbox', 'Partial Differential Equation Toolbox', 'Robust Control Toolbox', 'Signal Processing Toolbox', and 'Statistics Toolbox'. The main content area is titled 'Getting Started' and contains the following text: 'The MATLAB® high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. You can watch the [Getting Started with MATLAB video demo](#) for an overview of the major functionality. If you have an active Internet connection, you can also watch the [Working in the Development Environment video demo](#), and the [Writing a MATLAB Program video demo](#). This collection includes the following topics:'. Below this text, there is a list of topics with brief descriptions: 'Introduction' (Describes the components of the MATLAB system), 'Matrices and Arrays' (How to use MATLAB to generate matrices and perform mathematical operations on matrices), 'Graphics' (How to plot data, annotate graphs, and work with images), 'Programming' (How to use MATLAB to create scripts and functions, how to construct and manipulate data structures), 'Data Analysis' (How to set up a basic data analysis), 'Creating Graphical User Interfaces' (Introduces GUIDE, the MATLAB graphical user interface development environment), 'Desktop Tools and Development Environment' (Information about tools and the MATLAB desktop), and 'External Interfaces' (Introduces external interfaces available in MATLAB software). At the bottom of the main content area, there is a link to a printable version (PDF) of the documentation: 'A printable version (PDF) of this documentation is available on the Web—[MATLAB Getting Started Guide](#). For tutorial information about any of the topics covered in this collection, see the corresponding sections in the MATLAB documentation. For reference information about MATLAB functions, see the [MATLAB Function Reference](#).' Below this, there is a feedback section with the text 'Was this topic helpful?' and two buttons: 'Yes' and 'No'. At the bottom right of the main content area, there is a link to the 'Introduction' page. The footer of the window shows the copyright information: '© 1984-2010 The MathWorks, Inc. - [Terms of Use](#) - [Patents](#) - [Trademarks](#) - [Acknowledgments](#)'. The taskbar at the bottom of the screen shows the 'Inicio' button, several open applications including 'Explora...', 'main.cpp - ...', 'Gmail - RV...', 'Intro\_Mat...', 'MATLAB 7...', 'Editor - C:\...', and 'Help', and the system clock showing '17:30'.

**Getting Started**

The MATLAB® high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. You can watch the [Getting Started with MATLAB video demo](#) for an overview of the major functionality. If you have an active Internet connection, you can also watch the [Working in the Development Environment video demo](#), and the [Writing a MATLAB Program video demo](#). This collection includes the following topics:

<a href="#">Introduction</a>	Describes the components of the MATLAB system
<a href="#">Matrices and Arrays</a>	How to use MATLAB to generate matrices and perform mathematical operations on matrices
<a href="#">Graphics</a>	How to plot data, annotate graphs, and work with images
<a href="#">Programming</a>	How to use MATLAB to create scripts and functions, how to construct and manipulate data structures
<a href="#">Data Analysis</a>	How to set up a basic data analysis
<a href="#">Creating Graphical User Interfaces</a>	Introduces GUIDE, the MATLAB graphical user interface development environment.
<a href="#">Desktop Tools and Development Environment</a>	Information about tools and the MATLAB desktop
<a href="#">External Interfaces</a>	Introduces external interfaces available in MATLAB software.

A printable version (PDF) of this documentation is available on the Web—[MATLAB Getting Started Guide](#). For tutorial information about any of the topics covered in this collection, see the corresponding sections in the MATLAB documentation. For reference information about MATLAB functions, see the [MATLAB Function Reference](#).

Was this topic helpful?

[Introduction](#)

© 1984-2010 The MathWorks, Inc. - [Terms of Use](#) - [Patents](#) - [Trademarks](#) - [Acknowledgments](#)

# Ayuda de Matlab

Tecleando **HELP** obtendremos la ayuda general de **Matlab**.

Si tecleamos la opción **HELP** "nombre de función" obtenemos ayuda sobre esa función:

Ej: `>> help mean`

**MEAN** Average or mean value.

For vectors, **MEAN(X)** is the mean value of the elements in X. For matrices, **MEAN(X)** is a row vector containing the mean value of each column. For N-D arrays, **MEAN(X)** is the mean value of the elements along the first non-singleton dimension of X.

**MEAN(X,DIM)** takes the mean along the dimension DIM of X.

Example: If `X = [1 2 3; 3 3 6; 4 6 8; 4 7 7];`

then `mean(X,1)` is `[3.0000 4.5000 6.0000]` and  
`mean(X,2)` is `[2.0000 4.0000 6.0000 6.0000].'`

Class support for input X:

float: double, single

See also `median`, `std`, `min`, `max`, `var`, `cov`, `mode`.

Overloaded methods:

`timeseries/mean`

`ProbDistUnivParam/mean`

Reference page in Help browser

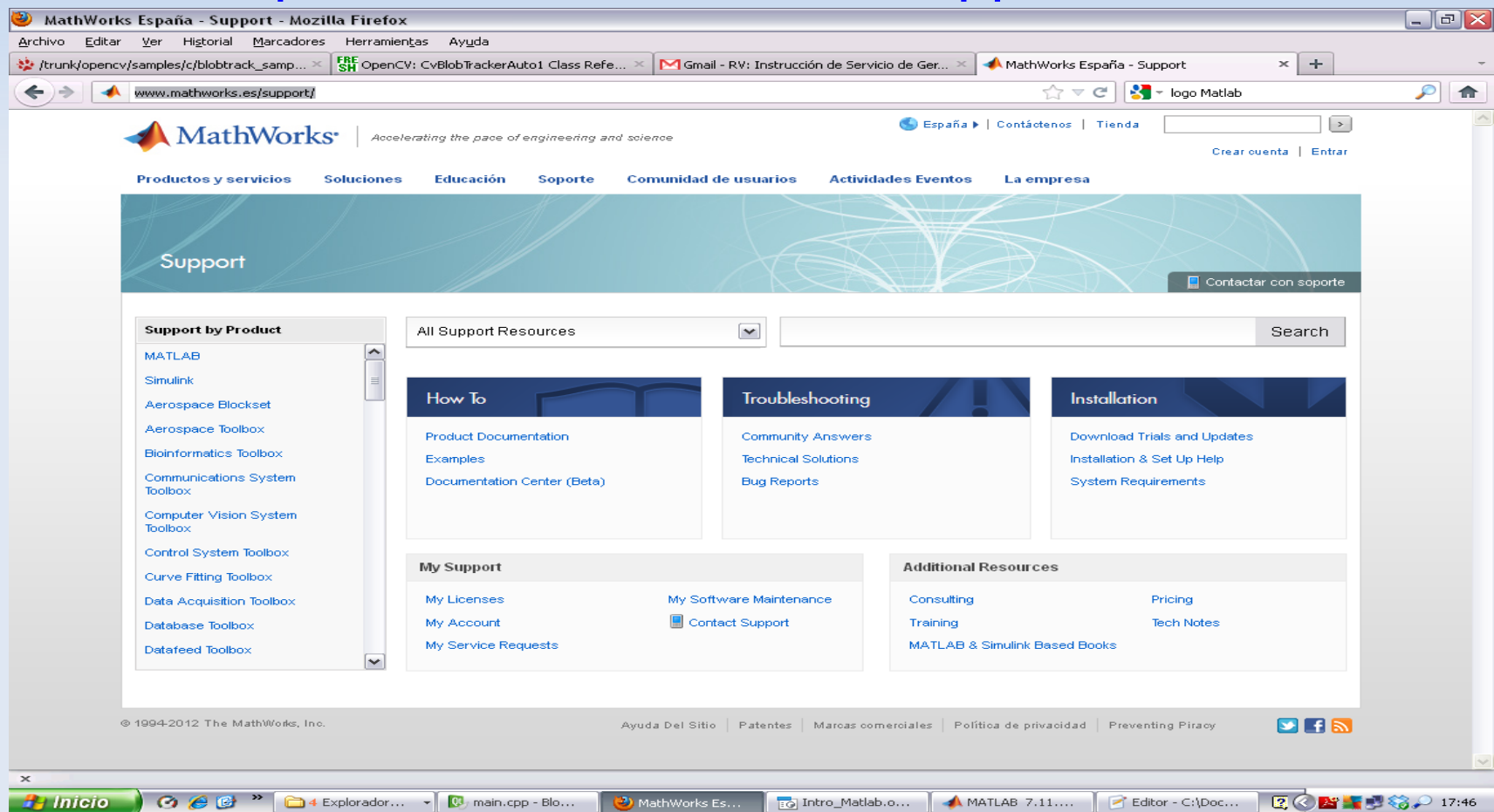
[doc mean](#)



# Ayuda de Matlab

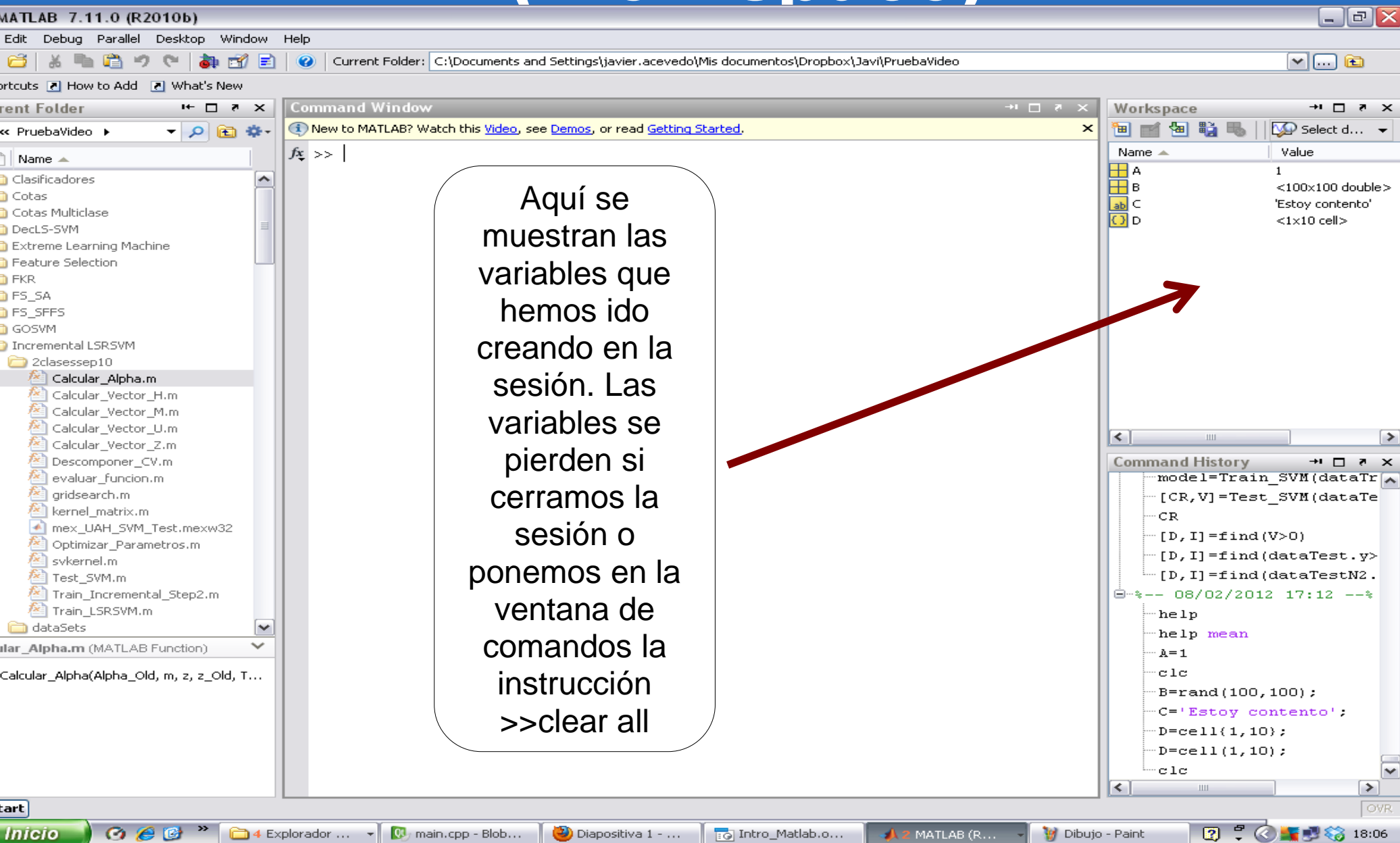
En la siguiente dirección se puede encontrar todo tipo de soporte para los diferentes productos de Matlab

<http://www.mathworks.es/support/>



# **HERRAMIENTAS DEL ESCRITORIO**

# Espacio de Trabajo (Workspace)



Aquí se muestran las variables que hemos ido creando en la sesión. Las variables se pierden si cerramos la sesión o ponemos en la ventana de comandos la instrucción `>>clear all`

Workspace

Name	Value
A	1
B	<100x100 double>
C	'Estoy contento'
D	<1x10 cell>

Command History

```
model=Train_SVM(dataTr  
[CR,V]=Test_SVM(dataTe  
CR  
[D,I]=find(V>0)  
[D,I]=find(dataTest.y>  
[D,I]=find(dataTestN2.  
--%-- 08/02/2012 17:12 --%  
help  
help mean  
A=1  
clc  
B=rand(100,100);  
C='Estoy contento';  
D=cell(1,10);  
D=cell(1,10);  
clc
```

# Histórico de comandos

The image shows the MATLAB 7.11.0 (R2010b) interface. The Command History window is open, displaying a list of commands executed. A red arrow points from a text box to the Command History window.

En el histórico de comandos se muestran las últimas instrucciones que se han metido en la ventana de comandos

Workspace:

Name	Value
A	1
B	<100x100 double>
C	'Estoy contento'
D	<1x10 cell>

Command History:

```
model=Train_SVM(dataTr
[CR,V]=Test_SVM(dataTe
CR
[D,I]=find(V>0)
[I,I]=find(dataTest.y>
[I,I]=find(dataTestN2.
-- 08/02/2012 17:12 --
help
help mean
A=1
clc
B=rand(100,100);
C='Estoy contento';
D=cell(1,10);
D=cell(1,10);
clc
```

# Histórico de comandos

The screenshot shows the MATLAB 7.11.0 (R2010b) environment. The 'Current Folder' pane on the left lists various files and subfolders. A red arrow points from a callout box to the 'Current Folder' pane. The callout box contains the text: 'Navegador. Permite seleccionar el directorio de trabajo'. The 'Command Window' is in the center, showing a prompt '>>'. The 'Workspace' pane on the right shows variables A, B, C, and D. The 'Command History' pane at the bottom right shows a list of commands executed, including 'model=Train\_SVM(dataTr', '[CR,V]=Test\_SVM(dataTe', 'CR', '[D,I]=find(V>0)', '[D,I]=find(dataTest.y>', '[D,I]=find(dataTestN2.', a date separator, 'help', 'help mean', 'A=1', 'clc', 'B=rand(100,100);', 'C='Estoy contento';', 'D=cell(1,10);', 'D=cell(1,10);', and 'clc'.

Navegador.  
Permite  
seleccionar el  
directorio de  
trabajo

Workspace

Name	Value
A	1
B	<100x100 double>
C	'Estoy contento'
D	<1x10 cell>

Command History

```
model=Train_SVM(dataTr
[CR,V]=Test_SVM(dataTe
CR
[D,I]=find(V>0)
[D,I]=find(dataTest.y>
[D,I]=find(dataTestN2.
-- 08/02/2012 17:12 --
help
help mean
A=1
clc
B=rand(100,100);
C='Estoy contento';
D=cell(1,10);
D=cell(1,10);
clc
```

# **Variables: Definición y Generación**

# Escalares

El elemento básico de trabajo de Matlab es la matriz. Sin embargo, vamos a definir algunos subtipos antes de trabajar con matrices:

Variable escalar: Se trata de una matriz 1x1 que tiene un valor (real o complejo).

Ejemplos: `>> a=1;`  
`>> b = 5 + i;`

¿Qué pasa en el Workspace cuando tecleamos esto?

Salvo que definamos lo contrario, las variables `i` y `j` están predefinidas (raíz de -1) y su uso denota que estamos trabajando con números complejos.

Ejemplo: `>> c=7+i;`  
`>> i=5;`  
`>> d=4+i;`

¿Qué ocurre con `d`? ¿Queda definida como un complejo?

# Escalares

Valores especiales:

. `[]` Con esta función declaramos una variable como vacía. En Matlab no es necesario declarar las variables ni reservar memoria para las mismas. Puede sernos útil cuando veamos concatenación.

. `Inf` / `-Inf` De esta forma queda declaradas las variables + infinito y – infinito.

Ej:

```
>> 1/0
```

. `NaN` (Not a number) es un mensaje que da matlab indicando que el resultado no se pudo determinar. Si una variable queda declarada como NaN, las variables que trabajen con esa se determinarán también como NaN

Ej:

```
>> b=0/0
```

```
>> f=b+3
```



# Vectores

Vectores: Un vector fila es una matriz de dimensiones  $1 \times N$  ( $N > 1$ ).

Un vector columna es una matriz de dimensiones  $N \times 1$ .

Para definir un vector fila debemos definir sus elementos entre corchetes separados por comas.

Ejemplos:

```
>> v = [1, 3, 5]
```

```
>> c = [4+i, 6, 6+2i, 8i];
```

Para definir un vector columna debemos definir sus elementos entre corchetes separados por punto y coma.

Ejemplos:

```
>> v2 = [3; 5; 6; 7];
```

```
>> r = [5; 7; 9; 0];
```

Observar en el espacio de trabajo cómo quedan definidas las variables anteriores.

# Vectores

## Operador dos puntos (:)

Podemos crear un vector fila cuyos elementos estén separados de forma uniforme invocando la siguiente instrucción:

Ejemplos:      Vector = valor inicial: incremento: valor final

```
>>V = 1:2:11
```

```
>>D = 4:3:25
```

```
>>H= 1:2:20
```

¿Qué sucede en este último caso? ¿llegamos hasta el valor final?

Si no lo indicamos, el valor del incremento por defecto es +1.

Ejemplo:

```
>> a=1:10
```

También podemos usar incrementos negativos.

```
G= 30:-1:5;
```

```
J= 1:-1:4;
```

¿Qué da en este último caso?

# Matrices

Tanto los escalares como los vectores pueden ser considerados casos particulares de las matrices.

Para crear una matriz debemos indicar primero los elementos de la primera fila, separados por comas. Para definir una nueva fila, incluimos un punto y coma. Todas las filas deben tener el mismo número de elementos:

Ejemplo: (Matriz 3x4 3 filas y cuatro columnas)

```
>> A=[1,3,6,7; 2,4,8,6; 0,9,1,3]
```

A =

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

Crear las siguientes matrices

$$M = \begin{bmatrix} 1 & 4 \\ 6 & 9 \\ 2 & 8 \end{bmatrix}$$

$$C = \begin{bmatrix} i & 2+6i & 9 & 4 \\ 2+i & 3+i & 2 & 7 \end{bmatrix}$$

# Matrices

Podemos definir matrices de ceros y unos mediante las instrucciones zeros y ones.

Ejemplos:

(Matriz 4x4 todo ceros)

```
>> M=zeros(4,4)
```

M =

```
0  0  0  0
0  0  0  0
0  0  0  0
0  0  0  0
```

(Vector fila de 1x 6 todo ceros)

```
>> V=zeros(1,6)
```

V =

```
0  0  0  0  0  0
```

(Matriz 2 x 4 todo unos)

```
>> B=ones(2,4)
```

B =

```
1  1  1  1
1  1  1  1
```

(Vector columna de cinco elementos todo uno)

```
>> W = ones (5,1)
```

W =

```
1
1
1
1
1
```

# Matrices

Podemos definir aleatoriamente los elementos de una matriz mediante las funciones `rand` y `randn`

Ejemplos:

(Matriz de valores aleatorios de 8 filas por 9 columnas, con valores distribuidos uniformemente entre 0 y 1)

```
>>R=rand(8,9)
```

R =

0.8241	0.2816	0.1115	0.4104	0.2076	0.0806	0.7444	0.2892	0.3055
0.2182	0.0068	0.5793	0.2375	0.3821	0.0433	0.4168	0.0731	0.1549
0.0996	0.4959	0.8704	0.4890	0.6603	0.4912	0.9074	0.1946	0.5555
0.6195	0.9885	0.6898	0.8061	0.7584	0.4466	0.0943	0.4175	0.7905
0.1038	0.7379	0.2430	0.3778	0.1731	0.4868	0.1813	0.2929	0.4439
0.7991	0.3107	0.3427	0.5180	0.5174	0.1659	0.9466	0.7021	0.9958
0.9029	0.6004	0.5454	0.0946	0.9953	0.3607	0.1008	0.2397	0.4366
0.3125	0.7817	0.0676	0.9091	0.7076	0.8807	0.3880	0.9595	0.3044

(Matriz de valores aleatorios de 4 filas por 4 columnas, con valores distribuidos según distribución normal)

```
>> R2=randn(4,4)
```

R2 =

-0.4830	1.5833	0.4616	1.5942
-0.1022	1.8838	0.4029	-0.8350
0.6043	-0.7290	1.3850	0.0803
-0.7471	-0.3469	-0.0714	0.3235

# Estructura

Una estructura es un contenedor de varias variables, a las cuales se accede mediante la llamada a la estructura seguida de un punto.

Ejemplo:

Para manejar datos relativos a una imagen podríamos hacerlo de la siguiente manera:

```
ancho=512;  
alto=256;  
Nombre_Imagen='img001.jpg'
```

Pero si queremos manejar varias imágenes no parece aconsejable estar declarando estas variables. En su lugar declaramos:

```
Imagen1.ancho=512;  
Imagen1.alto=256;  
Imagen1.Nombre_Imagen='img001.jpg'
```

VENTAJA: PROGRAMACIÓN MÁS ESTRUCTURADA

# CELDA

Las Celdas son arrays contenedores que pueden tener diferentes tipos de variables en cada uno de sus elementos.

```
A=cell{2,2}
```

Crea un array de dos celdas por dos celdas. Cada elemento puede ser diferente:

```
Texto='Hola, esto es una prueba';  
D.campo1=52;  
D.campo2=rand(23,32);  
D.campo3=ones(1,11);  
Matriz1=zeros(14,14);  
Matriz2=[];  
A{1,1}=Texto;  
A{1,2}=D;  
A{2,1}=Matriz1;  
A{2,2}=Matriz2;
```

Mediante celdas podemos crear también arrays de estructuras.

# Cargar y Salvar Variables

Supongamos que llevamos un buen rato trabajando. Queremos cerrar Matlab y nos gustaría guardar las variables, como por ejemplo los resultados que hayamos obtenido.

La forma más sencilla de hacerlo sería seleccionando las variables en el espacio de trabajo y con el botón derecho pulsar Guardar como...

Las variables se almacenan en un fichero .mat

Para recuperarlas, es suficiente con hacer un doble click en la ventana de exploración sobre el fichero que quedamos cargar.

La alternativa es utilizar los comandos save y load.



# **Concatenación e indexación**

# Concatenación

Si queremos concatenar horizontalmente separamos por comas (obligatorio: deben tener el mismo número de filas).

Si queremos concatenar verticalmente separamos por puntos y comas. (obligatorio: deben tener el mismo número de columnas)

Ejemplos:

```
>> M = [1,2,3; 4,5,6];  
>> M2 = [3,2,1;6,5,4];  
>> M3 = [M,M2]
```

M3 =

1	2	3	3	2	1
4	5	6	6	5	4

```
>> V=zeros(1,6)  
>> V2 = ones (1,3)  
>> V3 =[V,V2]
```

V3 =

0	0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---	---

```
>> M=[1,2,3;4,5,6];  
>> M2=[1,1,1;2,2,2;3,3,3];  
>> M3=[M;M2]  
M3 =
```

1	2	3
4	5	6
1	1	1
2	2	2
3	3	3

# Concatenación

Ejemplos:

Obtener las siguientes matrices (sin teclear todos los elementos):

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & -3 & -1 & 1 & 3 & 5 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

# Indexación

Para poder acceder al valor de un elemento de una matriz:

$a=M(f,c)$  donde  $f$  es el número de fila y  $c$  el número de columna.

El primer elemento siempre es el uno.

Ejemplo

$M=[0,4,5;3,2,1;8,6,4];$

$a=M(2,3);$

¿Qué valor tendría  $a$ ?

También podemos seleccionar una submatriz definiendolo de la siguiente forma:

$S=M(fi:ff,ci:cf)$  donde  $fi$ = fila inicial,  $ff$  = fila final,  $ci$  = Columna inicial,  $cf$ =columna final

Ejemplo

```
>>M = [2,3,4,5,6;  
       0,2,4,5,7;  
       1,4,6,7,9];  
>>M2= M(1:2,1:3)
```

# Indexación

Si solo ponemos el operador :, significa que queremos todos los elementos de esa fila o columna

Ejemplos

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

M =

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

```
>> M2=M(:, 1)
```

M2 =

4
1
-1

```
>> M3= M (2, :)
```

M3 =

1	2	3	4	5	6
---	---	---	---	---	---

# Indexación

También podemos hacer que los índices sean una definición de array o incluso un vector o matriz. Si indicamos en un vector la variable end quiere decir que tome hasta el final.

Ejemplo: `Z=zeros(3,3), ones(3,2);ones(2,5);`

`Z2=Z(4:end,:);`

`Z =`

0	0	0	1	1
0	0	0	1	1
0	0	0	1	1
1	1	1	1	1
1	1	1	1	1

`Z2 =`

1	1	1	1	1
1	1	1	1	1

Ejemplo: Crear una matriz aleatoria de 50 x 50 y obtener cada una de las submatrices 25x25 existentes.

# Indexación

Para poder acceder al valor de un elemento de una matriz:

$a=M(f,c)$  donde  $f$  es el número de fila y  $c$  el número de columna.

El primer elemento siempre es el uno.

Ejemplo

$M=[0,4,5;3,2,1;8,6,4];$

$a=M(2,3);$

¿Qué valor tendría  $a$ ?

También podemos seleccionar una submatriz definiendolo de la siguiente forma:

$S=M(fi:ff,ci:cf)$  donde  $fi$ = fila inicial,  $ff$  = fila final,  $ci$  = Columna inicial,  $cf$ =columna final

Ejemplo

```
>>M = [2,3,4,5,6;
```

```
       0,2,4,5,7;
```

```
       1,4,6,7,9];
```

```
>>M2= M(1:2,1:3)
```

# **OPERADORES ARITMÉTICOS**



# Operadores Suma y Resta

En un vector o matriz los operadores suma o resta actúan elemento a elemento. Las matrices o vectores deben tener la misma dimensión.

Ejemplo:

```
A=ones(3,3);  
B= eye(3); % Matriz identidad 3x3  
F=A+B
```

F =

```
2    1    1  
1    2    1  
1    1    2
```

Ejemplo:

```
A=[2+i, 3+2i;0, 1+i];  
B=[5+2i, 2; 1+i, 2+2i];  
F= A+B
```

F =

```
7.0000 + 3.0000i  5.0000 + 2.0000i  
1.0000 + 1.0000i  3.0000 + 3.0000i
```

Si sumamos un escalar con una matriz o vector sí queda admitido.

```
>> R=rand(5,5); a=5;
```

```
>> G= R + a
```

# Operador Multiplicación y División

Si utilizamos el operador \* estamos indicando que queremos usar una multiplicación matricial

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} =$$
$$\begin{pmatrix} a_{11}.b_{11} + a_{12}.b_{21} + a_{13}.b_{31} & a_{11}.b_{12} + a_{12}.b_{22} + a_{13}.b_{32} & a_{11}.b_{13} + a_{12}.b_{23} + a_{13}.b_{33} \\ a_{21}.b_{11} + a_{22}.b_{21} + a_{23}.b_{31} & a_{21}.b_{12} + a_{22}.b_{22} + a_{23}.b_{32} & a_{21}.b_{13} + a_{22}.b_{23} + a_{23}.b_{33} \\ a_{31}.b_{11} + a_{32}.b_{21} + a_{33}.b_{31} & a_{31}.b_{12} + a_{32}.b_{22} + a_{33}.b_{32} & a_{31}.b_{13} + a_{32}.b_{23} + a_{33}.b_{33} \end{pmatrix}$$

Si hacemos  $A*B$ , A debe tener el mismo número de columnas que el número de filas de B.

$C=A*B$  con  $A(M \times N)$  y  $B(N \times Z)$  tendremos  $C(M \times Z)$

```
Ejemplo: >>A=ones(1,10);  
         >>B=ones(1,10);  
         >> C= A*B
```

¿Por qué da error?

# Multiplicación y división por escalar

Si multiplicamos o dividimos una matriz (o vector) por un escalar, todos los elementos quedan multiplicados por ese escalar

Ejemplo:

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

A =

5	5	5
5	5	5
5	5	5

```
>> Ejemplo:
```

```
>>B=A/3
```

B =

1.6667	1.6667	1.6667
1.6667	1.6667	1.6667
1.6667	1.6667	1.6667

Utilizando las funciones de multiplicación y suma por un escalar, obtener un vector 1x10 con valores aleatorios con distribución uniforme de media nula y rango desde -1 hasta +1.

# Multiplicación y división elemento a elemento

Si lo que queremos es multiplicar cada uno de los elementos de dos vectores o matrices de iguales dimensiones tendremos que usar los operadores `.*` y `./`

>>Ejemplos

```
>>A=2*ones(3,3);
```

```
>>B=[1:3;2:4;3:5];
```

```
>> C= A.*B
```

$$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$
$$\begin{bmatrix} 2 & 4 & 6 \\ 4 & 6 & 8 \\ 6 & 8 & 10 \end{bmatrix}$$

```
>> V=2*ones(1,10)
```

```
>>V2=3*ones(1,10)
```

```
>>V.*V2
```

# Potencia

Ocurre lo mismo que con la multiplicación y la división.

```
>>V=2*ones(2,2)
```

```
>>V^2
```

Esta operación en realidad hace  $V \cdot V$  (producto matricial)

$$\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 8 & 8 \\ 8 & 8 \end{bmatrix}$$

Si lo que queremos es elevar cada miembro al cuadrado, podemos usar el operador  $\text{.^}$

Ejemplo:

```
>> A=2*ones(2,2);
```

```
>> A.^2
```

```
ans =
```

```
4 4
4 4
```

# **OPERACIONES MATEMÁTICAS BÁSICAS.**

# Operaciones Matemáticas

Permiten realizar operaciones sobre cada uno de los elementos de una matriz o vector:  
(Para los siguientes ejemplos tomamos x como un escalar, vector o matriz)

**sin(x)** calcula el seno de los elementos de x

**.cos(x)** calcula el coseno de los elementos de x

**.tan(x)** calcula la tangente de los elementos de x.

**.log(x)** calcula el logaritmo neperiano de los elementos de x.

**.log10(x)** calcula el logaritmo decimal de los elementos de x.

**.exp(x)** calcula el exponencial de los elementos de x.

**.asin(x)** calcula el arcoseno de los elementos de x.

**.acos(x)** calcula el arcocoseno de los elementos de x.

**.atan2(x)** calcula la arcotangente de los elementos de x.

# Operaciones Matemáticas

• **Sqrt (x)** Raiz cuadrada de cada uno de los elementos del vector x.

• **abs(x)** calcula el módulo de los elementos de x. Si todos los elementos de x son reales el resultado coincide con x. Solo tiene sentido si trabajamos con complejos.

• **angle(x)** calcula la fase de los elementos de x. Si todos los elementos de x son reales devuelve todo ceros. Al igual que el caso anterior solo tiene sentido si trabajamos con complejos.

• Ejemplos:

Calcular el módulo y la fase de esta serie de complejos

$$z_1=5+3i \quad z_2= 4-2i \quad z_3= 3-2i \quad z_4= 5 +1.3i$$

Calcular el seno del siguiente vector de ángulos

$$A=[ \pi/2, \pi/4, \pi/3, \pi, 3\pi ]$$



# **OPERACIONES SOBRE VECTORES Y MATRICES.**

# Operaciones sobre vectores y matrices

A diferencia de las operaciones aritméticas, en las que se consideraba cada elemento aislado, cada operación afecta al conjunto de los elementos. En el caso de matrices, pueden operar por filas o por columnas.

**.Sum** Suma de los elementos de un vector

sum(v) Ejemplo: `>> A=1:10; b=sum(A) ans= 55 (1+2+3+..+10)`

¿Cuánto suman los números pares hasta 500?

Si A es una matriz, sum(A), nos da las sumas de los elementos de cada columna. Si queremos los elementos de cada fila, tenemos que ejecutar sum(A,2)

Ejemplo:

```
>> A=[1:5;1:5]
```

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

```
ans =
```

```
    2    4    6    8   10
```

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

```
ans =
```

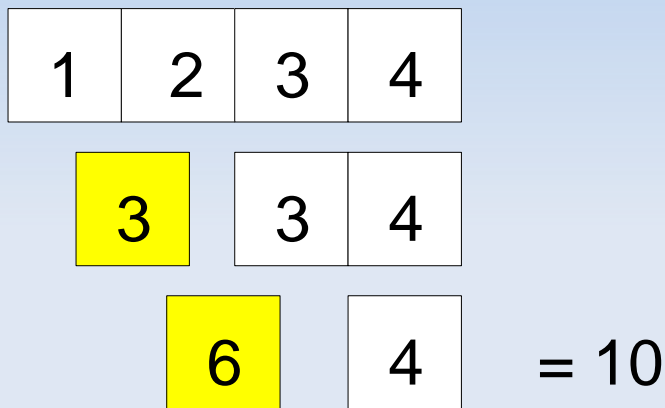
```
    15
```

```
    15
```

# Operaciones sobre vectores y matrices

**.cumsum** Suma de los elementos de un vector de forma acumulativa.

.V=1:4; cumsum(V) sería equivalente a efectuar las siguientes operaciones



Ejemplo:

```
>> V=1:4; cumsum(V)
```

```
ans =
```

```
1    3    6   10
```

Para matrices, cumsum(A) calcula la suma acumulativa de las columnas, si queremos hacerlo en la dirección de las filas debemos ejecutar cumsum(A,2)

# Operaciones sobre vectores y matrices

• **mean(x)** Calcula la media aritmética de los elementos de un vector. Si se trata de una matriz, calcula la media de las columnas.

•>> Ejemplo: `A=[2*ones(1,5), 5*ones(1,10)]; mean(A)`

•ans 4

• **length (x)** Devuelve la longitud (número de elementos) del vector x.

• **size (A)** Devuelve las dimensiones de la matriz A. `size(A,1)` devuelve el número de filas, mientras que `size(A,2)` devuelve el número de columnas.

•Ejemplo: Calcular la media aritmética de un vector sin recurrir a la función mean

•>> `sum(v)/length(v)`

•Ejemplo: Calcular la norma 2 de un vector

•>>`sqrt(sum(v.^2))`

# Operaciones sobre vectores y matrices

**.sort(x,dim, mode)** Ordena la matriz o vector x a lo largo de la dimensión indicada en dim. Si x es un vector, el valor de dim debe ser 2. Mode puede ser 'ascend' o 'descend'. Devuelve dos arrays, los índices de posición que tenían los elementos y el valor de los mismos.

.Ejemplos:

```
.>> V=rand(2,10);
```

```
.>> [D,I]=sort(V,2,'ascend');
```

```
.>> Ejemplo: Calcular la media de los 5 valores mayores de un vector aleatorio
```

```
.>> V=rand(1,50);
```

```
.>> [D,I]=sort(V,2,'ascend');
```

```
.>> mean(D(1:5))
```

**.find (condición)** Devuelve los valores y los índices de los elementos que cumplen una condición dada. Si no se indica nada más que el vector o la matriz, se supone que la condición es buscar los elementos que sean mayores que cero.

.%Generamos una matriz con distribución uniforme entre -1 y 1

```
.>> B=2*rand(3,3)-1;
```

```
.>> [I,J, V]=find(B>0.5)
```

.I contiene índices de las filas, J de las columnas y V los valores de las mismas.

# Operaciones sobre vectores y matrices

**. inv(X) Inversa de una matriz .**

Para que la función devuelva un valor coherente, X debe ser una matriz cuadrada no singular, es decir, que exista el inverso.

**Ejemplo:**

**V=[5;0;0];**

**Z=[1+i, 5, 3+2\*i;**

**5, 2+2\*i, 10;**

**3+2\*i, 10, 1+3\*i];**

**I=inv(Z)\*V**

# **CONTROL DE FLUJO.**

# Control de Flujo

## Ejecución condicional (If)

Si queremos que un código se ejecute únicamente si sucede una condición, utilizaremos las secuencias de control:

```
If <<condición>>
```

```
    Código a ejecutarse si sucede la condición
```

```
end
```

Ejemplo:

```
A=10*rand(1,10);
```

```
B=5;
```

```
If (A(1)>=5)
```

```
    B=1;
```

```
end
```



# Control de Flujo

## Condiciones

- == igual
- >= mayor o igual
- <= menor o igual
- < menor
- > mayor
- ~=
- lseempty (variable) – TRUE si la variable está vacía.
- En general cualquier función se considerará como condición y ejecutará el código si devuelve 1 no ejecutando si devuelve 0.
- Simultaneidad de condiciones (&)
- ejemplo if(A==5 & B>=3)
- Que ocurra una u otra (||) if(A==5 || B>=3)

# Control de Flujo

Si lo que queremos es que se ejecute un código si se cumple una condición y otro diferente si la condición no se cumple:

If <<condición>>

Código a ejecutarse si sucede la condición

else

Código a ejecutarse si no sucede la condición

end

Podemos anidar condiciones

If (A==3)

%Código a ejecutarse si A es igual a 3

C=4;

if(B~=2)

%Código a ejecutarse si B es distinto

de 2. Para ejecutar este código, es necesario que A sea igual a 3

C=7;

end

else %si A no es igual a 3

C=5;

end

# Control de Flujo

## Secuencia For

Cuando queremos que un determinado código se repita varias veces cada vez con una variable diferente.

Ejemplo:

%Queremos sumar a la variable A los 5 números pares más pequeños de un array B

```
A=0;  
B=round(20*rand(1,15));  
F=sort(B,'ascend');  
A=A+F(1);  
A=A+F(2);  
A=A+F(3);  
A=A+F(4);  
A=A+F(5);  
A=0;  
For i=1:5  
A=A+F(i);  
end
```

# Control de Flujo

%Ejemplo:

Crear dos secuencias A y B de números aleatorios enteros del 0 al 30 ordenados de menor a mayor. El tamaño de estas secuencias también debe ser aleatorio con tamaños desde el 1 hasta el 40.

Se trata de calcular las posiciones de A y B que cumplan que la diferencia entre ellos en valor absoluto es menor y en igualdad del valor anterior diferenciar por aquella pareja en la que la suma entre los números es mayor.

%Variante del programa anterior.

Mismos dos arrays, pero se trata de buscar aquellas posiciones para las que la diferencia absoluta se considera sumando los vecinos a la izquierda y derecha de la posición actual en cada array. Para simplificar el programa, descartaremos los extremos.

# Control de Flujo

Secuencia while

Se ejecuta el código mientras se cumple una condición

%Ejemplo

```
A=5;
```

```
while (A>2)
```

```
A=round(4*rand(1,1));
```

```
End
```

%Solo saldremos cuando se produzca la condición  $A > 2$

Ojo!!! Intentar evitar siempre que sea posible... si no aseguramos que la condición se rompe en algún momento puede dejar el ordenador colgado.

# Resumen

- Matlab nos permite ejecutar complicadas operaciones de forma muy sencilla.
- Idóneo para resolver muchos problemas matemáticos.
- Ofrece muchísimas más posibilidades que las que hemos comentado.