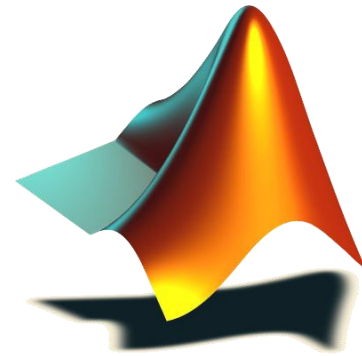
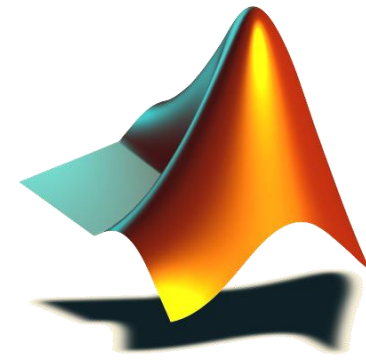


Introducción a MATLAB

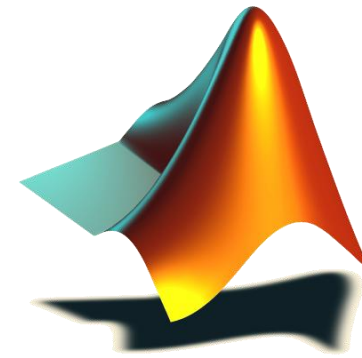


Índice



- Elementos básicos de MATLAB
- Uso del Help
- Formatos y variables
- Uso de las funciones elementales
- Los M-archivos
- Programación en MATLAB
- Matrices y vectores
- Gráficas en MATLAB

Elementos básicos de MATLAB

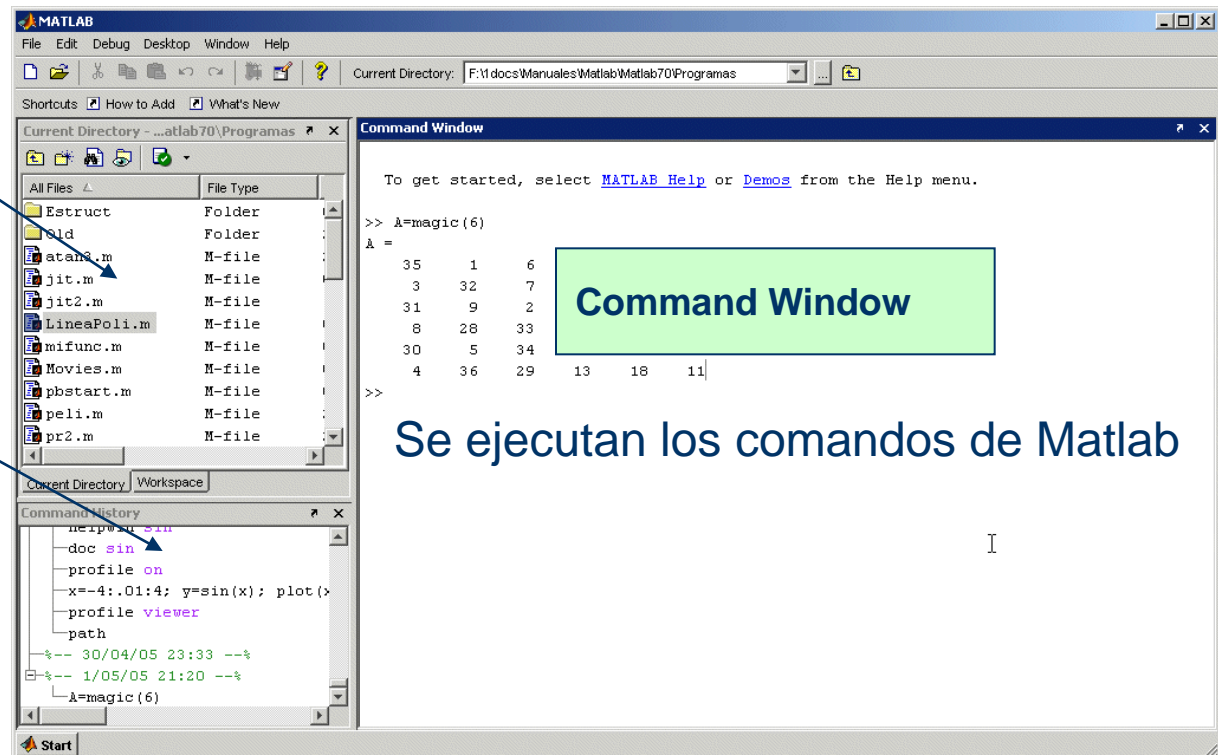


Eligiendo la opción **Desktop /Desktop Layout/Default**
se abre una ventana similar a

Current Directory/workspace:

Command History:

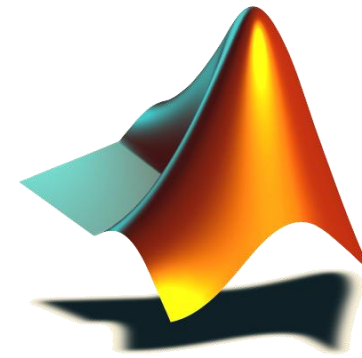
Start (función análoga a
la de Inicio de Windows)

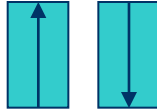


Command Window

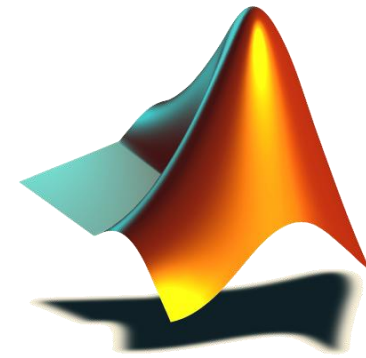
Se ejecutan los comandos de Matlab

La ventana de comandos (Command Window)



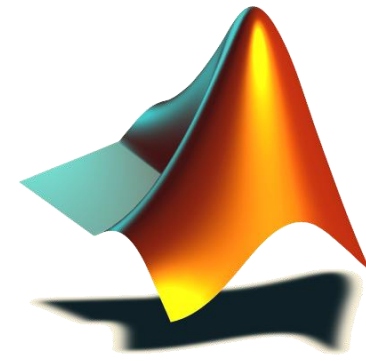
- Es donde se ejecutan las órdenes. Pueden ejecutarse también archivos con la extensión *.m que tengamos definidos.
- Muestra los errores que se producen en los *.m y aparece un subrayado que marca la línea donde se ha producido el error. Clicando en él se accede a la línea del fichero por medio del **Editor/Debugger**.
- >> (prompt) nos indica que el programa está a la espera de nuestras instrucciones.
- **Return** para ejecutar instrucciones.
- “... “ para escribir una instrucción de más de una línea.
- Las flechas para recuperar las órdenes previas. 
- Autocompleta el nombre de una función usando la tecla **Tab**.

La ventana de comandos (Command Window)



- Permite acceder a la página help clicando con el botón derecho sobre una función.
- El comando **diary** ('file') ordena a MATLAB que grabe todas la operaciones que se realizan en su ventana y que guarde los resultados en el archivo de texto de nombre 'file'. Al escribir **diary on** y **diary off** activa y desactiva la grabación.
- **clear** para limpiar las variables y las funciones de la memoria.
- **clc** para limpiar la pantalla, elimina todas las salidas anteriores.
- **home** para llevar el prompt (>>) a la primera línea de la ventana, manteniendo todas las salidas anteriores.
- Para salir de Matlab: **quit** o **exit**.
- Para comentar: **%**.

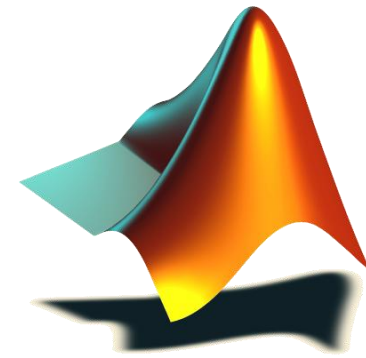
Histórico de comandos (Command History)



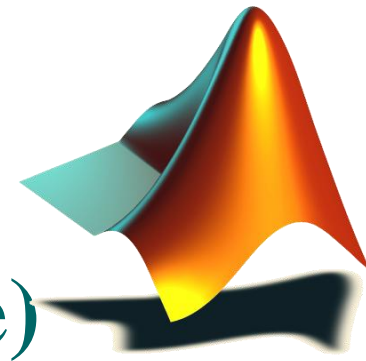
- Muestra los últimos comandos ejecutados en Command Window.
- Permite recuperar las instrucciones anteriores y ejecutarlas (mediante doble clic o copiándolas y volcándolas en la ventana de comandos).

El directorio actual

(Current directory)

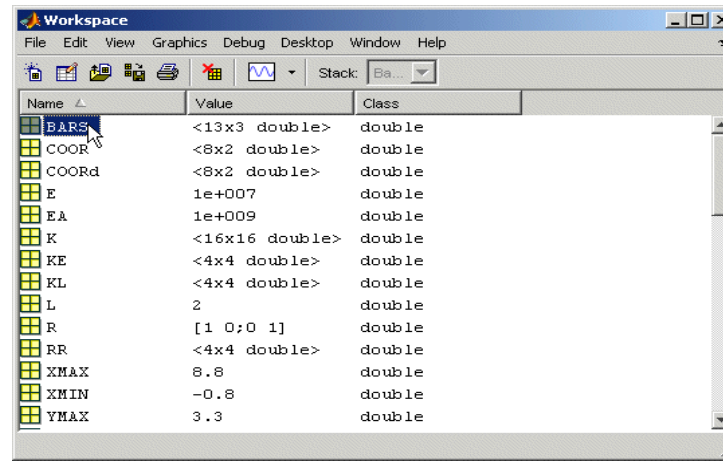


- Donde el usuario debe guardar los archivos, para que Matlab los detecte.
 - **pwd** muestra cuál es el directorio actual,
 - **dir** nos muestr ael contenido de dicho directorio,
 - **cd** nombre, cambia de directorio actual.
- Los ficheros *.m se ejecutan tecleando su nombre en la línea de comandos, pero para que un fichero *.m pueda ser ejecutado ha de cumplirse una de estas opciones:
 1. Que esté en el directorio actual.
 2. Que esté en uno de los directorios indicados en el Path.
- Los ficheros *.m que aparecen en el directorio actual pueden abrirse con el Editor/Debugger mediante doble clic.

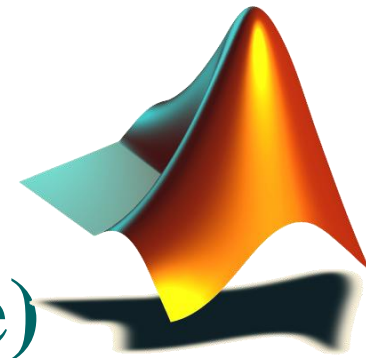


El espacio de trabajo (Workspace)

- Es el conjunto de variables que están definidas en la memoria de Matlab en un determinado momento.
- **Who** o **Whos** Son órdenes que nos permiten obtener información sobre el workspace. La segunda proporciona información más detallada.
- La ventana **Workspace** constituye un entorno gráfico para ver las variables definidas. Se activa con el comando **View/workspace**.



El espacio de trabajo (Workspace)

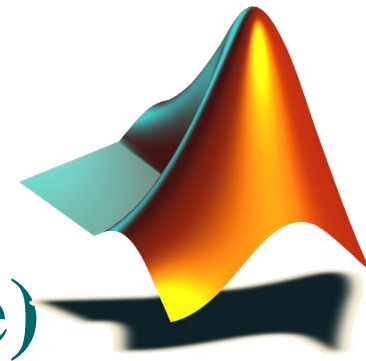


- Si hacemos doble clic en un elemento de la ventana de Worspace, aparece una nueva ventana **Array Editor** como por ejemplo:

Matriz 13x3

| Name | Value | Class |
|-------|----------------|--------|
| BARS | <13x3 double> | double |
| COORD | <8x2 double> | double |
| COORD | <8x2 double> | double |
| E | 1e+007 | double |
| EA | 1e+009 | double |
| K | <16x16 double> | double |
| KE | <4x4 double> | double |
| KL | <4x4 double> | double |
| L | 2 | double |
| R | [1 0;0 1] | double |
| RR | <4x4 double> | double |
| XMAX | 8.8 | double |
| XMIN | -0.8 | double |
| YMAX | 3.3 | double |

El espacio de trabajo (Workspace)

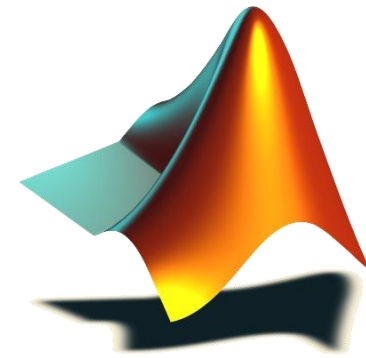


Array Editor para la matriz BARS

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|----|---|---|--------|---|---|---|---|
| 1 | 1 | 2 | 1e+009 | | | | |
| 2 | 1 | 3 | 1e+009 | | | | |
| 3 | 2 | 3 | 1e+009 | | | | |
| 4 | 2 | 4 | 1e+009 | | | | |
| 5 | 2 | 5 | 1e+009 | | | | |
| 6 | 3 | 5 | 1e+009 | | | | |
| 7 | 4 | 5 | 1e+009 | | | | |
| 8 | 4 | 6 | 1e+009 | | | | |
| 9 | 5 | 6 | 1e+009 | | | | |
| 10 | 5 | 7 | 1e+009 | | | | |
| 11 | 6 | 7 | 1e+009 | | | | |
| 12 | 6 | 8 | 1e+009 | | | | |
| 13 | 7 | 8 | 1e+009 | | | | |
| 14 | | | | | | | |
| 15 | | | | | | | |

- Es interesante saber que no sólo permite ver los valores de los elementos, sino también modificarlos clicando sobre la celda correspondiente.
- Se puede elegir el formato para visualizar los datos.

El explorador de caminos (Path Browser)

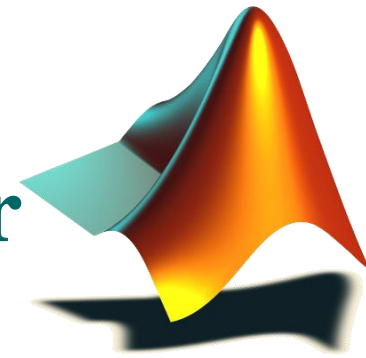


- Matlab puede llamar a funciones tanto propias como programadas por los usuarios. Puede haber funciones distintas con el mismo nombre.
- Es interesante saber cómo busca Matlab cualquier función y la clave está en el camino de búsqueda (search path) que el programa usa cuando encuentra el nombre de una función.
- Search path es una lista de directorios que se puede ver y modificar en la línea de comandos o mediante el cuadro de diálogos **File/Set Path**.

Si usamos por ejemplo nombre1 en un comando, el proceso que sigue search path para conocer nombre1 es:

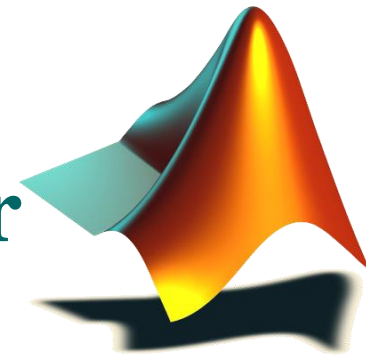
1. Ver si es una variable previamente definida por el usuario.
2. Ver si es una función interna de Matlab.
3. Ver si es una función creada por el usuario.
4. Etc.

El editor de ficheros y depurador de errores (Editor & Debugger)



- Los M-archivos son ficheros de texto ASCII con la extensión *.m que contienen conjuntos de comandos o definición de funciones.
- Al teclear el nombre de un fichero *.m en la ventana de comandos y pulsar Intro, se ejecutan uno tras otro todos los comandos contenidos en el fichero.
- Matlab dispone de un editor para crear y modificar estos archivos. Este editor muestra si algún comando contiene errores.
- Se puede comentar un conjunto de líneas seleccionadas usando el botón derecho del ratón y marcando la opción Comment (Uncomment para volver a la opción de código ejecutable).

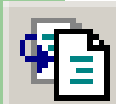
El editor de ficheros y depurador de errores (Editor & Debugger)



Set/Clear Breakpoint. Coloca o borra un ***breakpoint*** en la línea en que está el cursor.



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***.

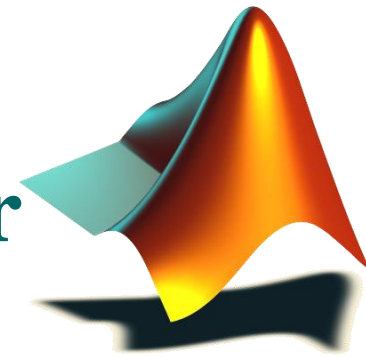


Quit Debugging. Terminar la ejecución del ***Debugger***.



Clear All Breakpoints. Elimina todos los ***breakpoints*** que haya en el fichero.

El editor de ficheros y depurador de errores (Editor & Debugger)



Si elegimos **Run** en el menú **Debug**, pulsando la tecla **F5**,

Breakpoint

Flecha Verde

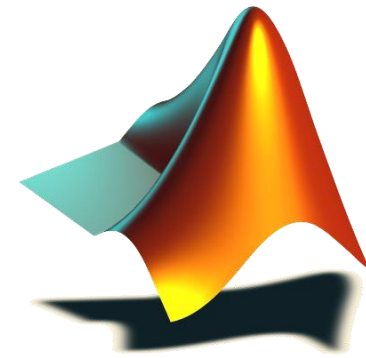
The image shows the MATLAB Editor & Debugger window. The script 'Prueba1.m' is open, showing the following code:

```
1 % fichero Prueba1.m
2 clear all
3 A=rand(3,3);
4 A=A';
5 C=inv(A);
6 D=C*A;
7 disp('
8
```

Line 3 has a red circle around the 'A' variable, indicating a breakpoint. Line 5 has a green arrow pointing to the 'C' variable, indicating the current execution point. A yellow tooltip is displayed over line 6, showing the value of 'A':

| | | |
|-----------------|-------------|-------------|
| A: 3x3 double = | | |
| 9.5013e-001 | 4.8598e-001 | 4.5647e-001 |
| 2.3114e-001 | 8.9130e-001 | 1.8504e-002 |
| 6.0684e-001 | 7.6210e-001 | 8.2141e-001 |

The status bar at the bottom shows 'Ln 5 Col 1 OVR'.

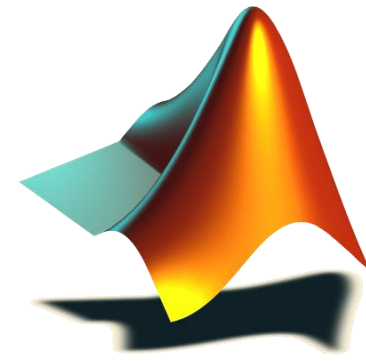


Uso del Help

Se puede acceder usando:

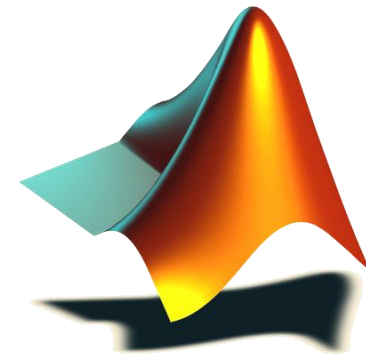
- **Help/Full Product Family Help.** Para buscar información general sobre MATLAB o productos de la familia.
- **Help/Matlab help.**
 - **Functions.** Informa sobre las funciones de Matlab.
 - **Handle Graphics.** Informa sobre propiedades de los elementos gráficos.
 - **Documentation Set.** Manuales del programa.
 - **Product Demos.** Colección de ejemplos programados que pueden ejecutarse y puede examinarse el código.
 - **What's New.** Novedades de la versión.
 - **Printing the Documentation Set.** Para abrir los manuales en PDF.
 - **The MathWorks Web Site Resources.** Informaciones disponibles en la Web de la empresa.

En la parte izquierda de la ventana, marcando Contents, aparece un índice en forma de árbol. El resto de pestañas son (Index), (Search) formulario de búsqueda y (Demos) para la colección de ejemplos.



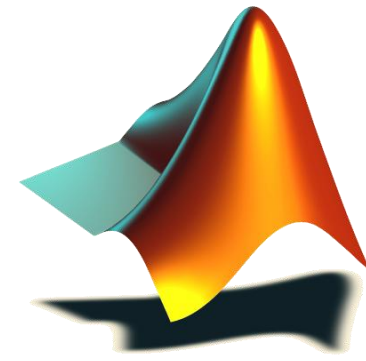
Uso del Help

- Help/Using the desktop. Para configurar el entorno de desarrollo o Desktop.
- Help/Using the Command Window.
- Desde la línea de comandos tecleando *help*.
- Comando *lookfor*.



Formatos y variables

- **Format long:** 16 dígitos.
- **Format short** o **format:** 4 dígitos.
- **Format long e:** 16 dígitos más exponente.
- **Format short e:** 4 dígitos más exponente.



Formatos y variables

Para definir una variable en MATLAB :

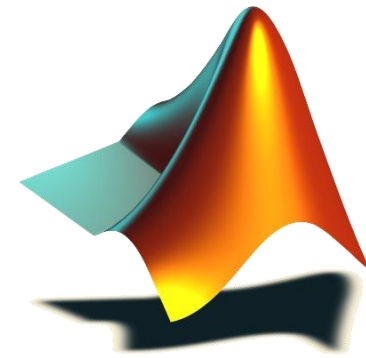
```
>>nombre_variable=expresión
```

Para saber el valor de una variable,

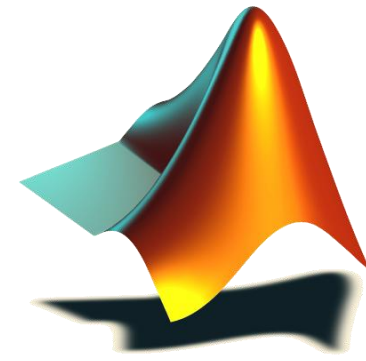
```
>>nombre_variable return
```

Por defecto se guarda en una variable llamada **ans.**

Uso de las funciones elementales



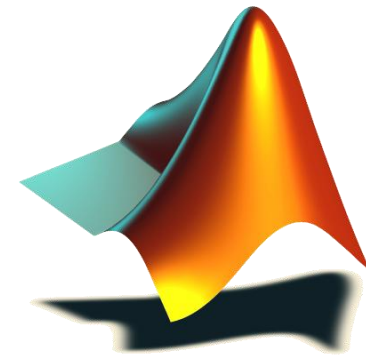
- **sin(x)**: seno de x
- **cos(x)**: coseno de x
- **tan(x)**: tangente de x
- **log(x)**: logaritmo neperiano de x
- **log10(x)**: logaritmo decimal de x
- **exp(x)**: funcion exponencial de x
- **sqrt(x)**: raíz cuadrada de x
- **abs(x)**: valor absoluto de x



Los M-archivos

Suelen dividirse en dos grandes grupos: archivos de instrucciones y archivos de funciones.

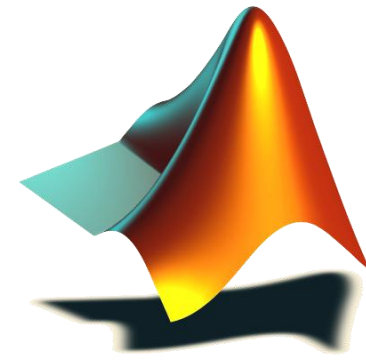
- Archivos de instrucciones:
 - Consiste en una sucesión de instrucciones de Matlab.
 - Para ejecutarlas, basta escribir su nombre en pantalla (sin la extensión) y pulsar Return
 - Son básicamente usadas para introducir datos en matrices de grandes dimensiones.
 - Cuando se ejecuta desde la línea de comandos, las variables creadas pertenecen al espacio de trabajo de base de Matlab, mientras que si se ejecuta desde una función, las variables pertenecen al espacio de trabajo de la función.
 - Conviene poner “;” al final de cada sentencia
 - El comando **echo** para imprimir los comandos a medida que se van ejecutando.



Los M-archivos

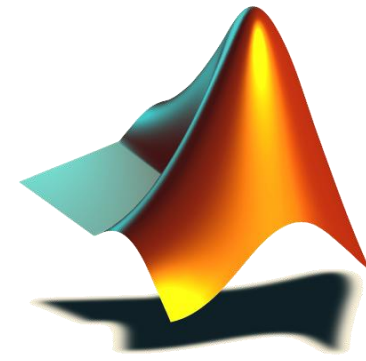
Suelen dividirse en dos grandes grupos: archivos de instrucciones y archivos de funciones.

- Archivos de funciones:
 - Existen bibliotecas de archivos que se venden o se distribuyen en internet.
 - Las variables en los archivos de funciones son locales.
 - Se aconseja que el nombre del archivo y de la función sea el mismo.
 - La estructura de este tipo de archivos es:
 - Primera línea: comienza con la orden
function [argumentos de salida] =nombreFuncion (argumentos de entrada)
 - Si no hay valores de retorno se omiten los corchetes y el signo igual
 - Si sólo hay un valor de retorno no hace falta poner corchetes.



Los M-archivos. Ejemplo 1

Ejemplo: definir la función *prueba* que devuelva $f(x)=x^3+3\cos(x)$.
Primero suponiendo que la entrada es un número y luego
suponiendo que la entrada sea un vector.



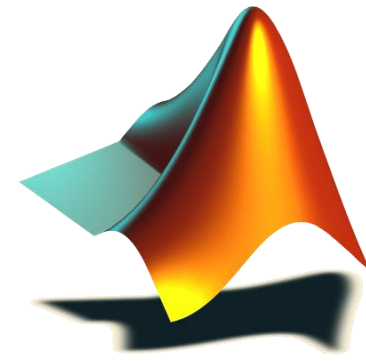
Los M-archivos. Ejemplo 1

Ejemplo: definir la función *prueba* que devuelva $f(x)=x^3+3\cos(x)$.

```
function y=prueba(x)
y=x^3+3*cos(x);
```

Si la entrada es un vector.

```
function y=prueba(x)
y=x.^3+3.*cos(x);
```



Los M-archivos

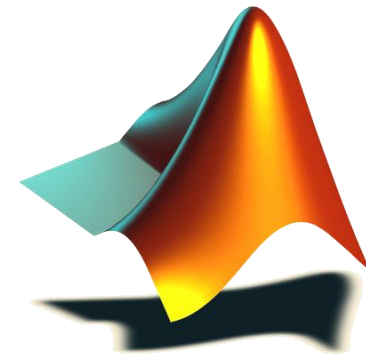
Los archivos *.m pueden llamar a otros *.m o llamarse a sí mismo de forma recursiva.

- Funciones con un número arbitrario de argumentos.

Se hace mediante la variable **varargin** que es un vector de celdas. No es necesario que esta variable sea la única de entrada, pero sí que debe ser la última. A los elementos de este tipo de variables se acceden mediante { }.

- Sub-Funciones.

Funciones adicionales introducidas en un mismo fichero. Sólo pueden ser llamadas por las funciones definidas en ese fichero.



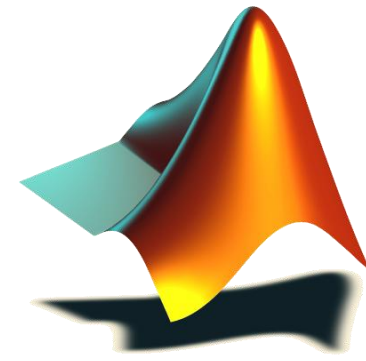
Los M-archivos. Ejemplo 2

Ejemplo de subfunción:

```
function y=mi_fun(a,b)
y=subfun1(a,b);
```

```
function x=subfun1(y,z)
x=subfun2(y,z);
```

```
function x=subfun2(y,z)
x=y+z+2;
```



Programación en MATLAB

- SENTENCIA IF.

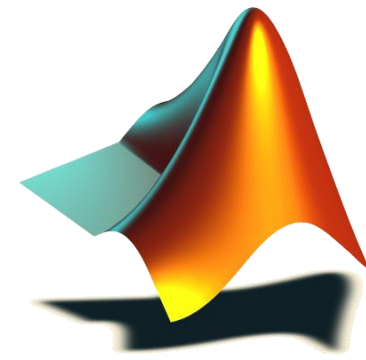
```
If condicion  
    sentencias  
end
```

```
If condicion1  
    bloque1  
elseif condicion2  
    bloque2  
else  
    bloque3  
end
```

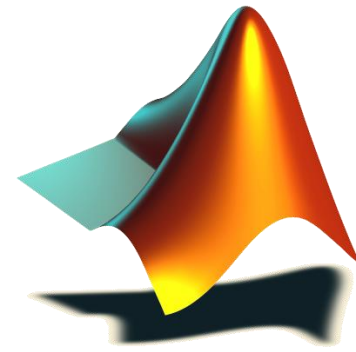
Operaciones lógicas:

>, <, >=, <=, == (igual)
| (or), & (and)
~ (no), ~= (no igual)

Ejemplo 3



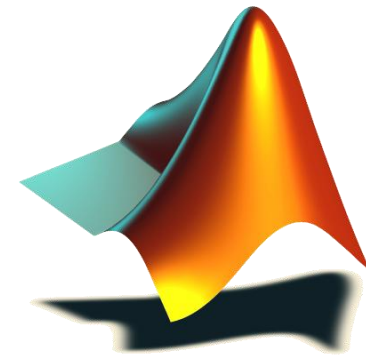
Diseñe una función *paridad* sobre los números enteros que asigna el valor 1 a los números pares positivos, el valor -1 a los impares positivos y el valor cero a los números restantes.



Ejemplo 3

Diseñe una función *paridad* sobre los números enteros que asigna el valor 1 a los números pares positivos, el valor -1 a los impares positivos y el valor cero a los números restantes.

```
function y=paridad(n)
if n<=0
    y=0;
elseif rem(n,2)==0
    y=1;
else
    y=-1;
end
```



Programación en MATLAB

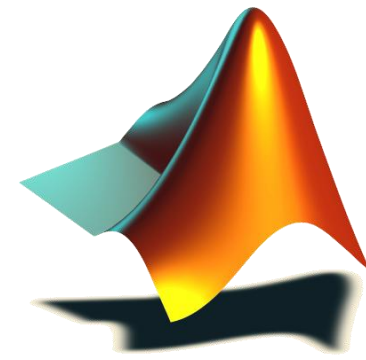
- SENTENCIA FOR.

```
for "variable"="vector"  
  "Instrucciones sobre la variable"  
end
```

bucles anidados

```
for i=1:n  
  for j=1:n  
    sentencias  
  end  
end
```

Ejemplo 4



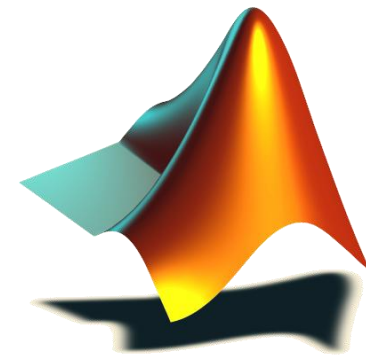
Diseñe una función *iter* que devuelva el término n-ésimo de la iteración

$$x_{(n+1)} = x_n^2 - 2x_n,$$

$$x_0 = 4.$$

Obtenga x_1 , x_5 y x_9 .

Ejemplo 4



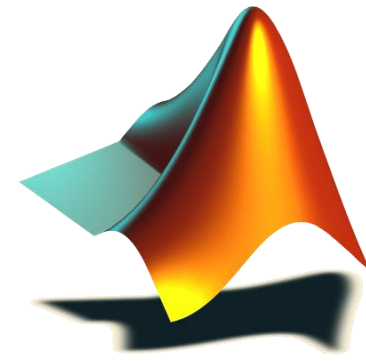
Diseñe una función *iter* que devuelva el término n-ésimo de la iteración

$$x_{(n+1)} = x_n^2 - 2x_n,$$

$$x_0 = 4.$$

Obtenga x_1 , x_5 y $x_9 \Rightarrow \text{iter}(1), \text{iter}(5), \text{iter}(9)$.

```
function y=iter(n)
x=4;
for k=1:n
    x=x^2-2*x;
end
y=x;
```

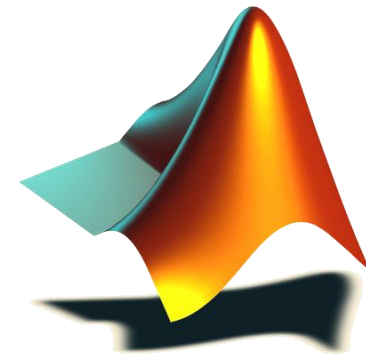


Programación en MATLAB

- SENTENCIA WHILE.

```
while "relación lógica (contador)"  
    "Instrucciones (contador)"  
end
```

- Break: Hace que termine el bucle for o while
- Continue. Hace que pase a la siguiente iteración del bucle for o while



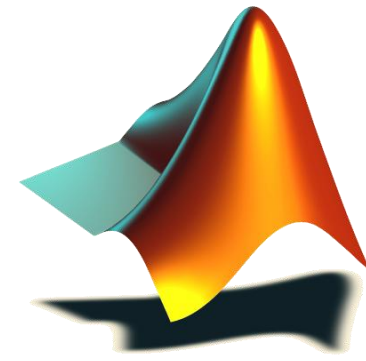
Matrices y vectores

- *Definición de matrices.*

Se sugiere que se usen las letras mayúsculas para matrices y las minúsculas para vectores y escalares

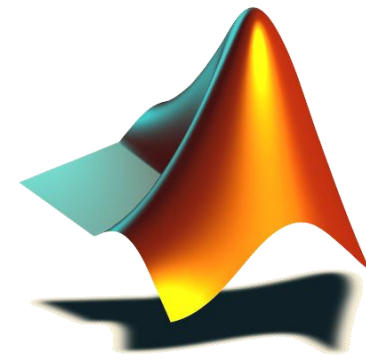
Las matrices pueden definirse de varias formas:

- **Introduciendo directamente los elementos**
 - Entre corchetes, colocando las filas una a continuación de otra separadas por “;”
 - Entre corchetes, colocando cada fila en un renglón.
 - La matriz vacía se representa por []
- **A partir de matrices predefinidas**
- **A partir de otras matrices**
 - Recibiendo propiedades, por composición de submatrices o bien modificándola.
- **Usando el operador “:”**



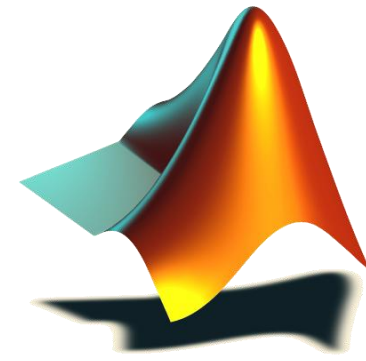
Matrices y vectores

- *Vectores:*
 - Si sólo introducimos una fila, tenemos un vector fila.
 - Si separamos cada elemento por “;” o introducimos los elementos en renglones distintos, tenemos un vector columna.
 - La estructura a:b:c crea un vector entre los números a y c incrementando cada coordenada con el número b



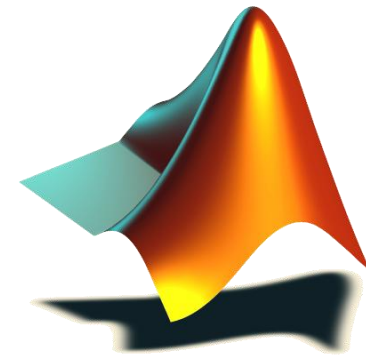
Matrices usuales

- Identidad de orden n : **eye(n)**
- Nula de tamaño $m \times n$: **zeros(m,n)**
- Matriz de unos: **ones(m,n)**
- Matriz aleatoria: **rand(m,n)**
- Matriz diagonal: **diag(v)**
- Matriz inversa: **inv(A)**
- Matriz traspuesta: **A'**



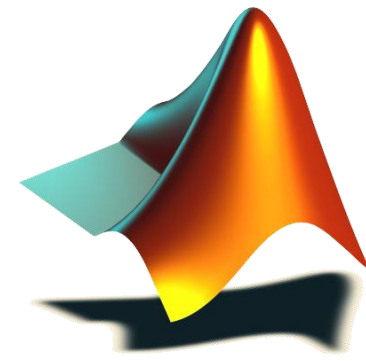
Operaciones con Matrices

- Suma y resta: $+$ $-$
- Producto: $*$ $.*$
- Potencia: $^$ $.^$
- Cociente izq.: $/$ $./$
- Cociente der.: \backslash $.\backslash$
- Transpuesta: $'$
- Orden: **size(A)**
- Traza: **trace(A)**
- Determinante: **det(A)**
- Diagonal de A: **diag(A)**
- Inversa: **inv(A)**
- Rango: **rank(A)**



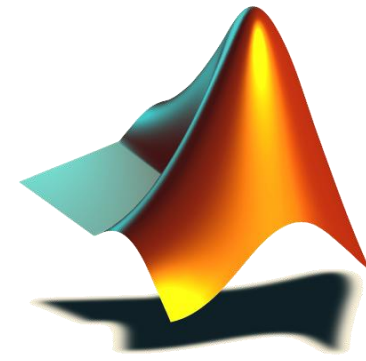
Vectores

- Edición
 - » **u = [1 2 3]** (Vector fila)
 - » **v = [1,2,3]** (Vector fila)
 - » **w = [1;2;3]** (Vector columna)
 - » **w = [1
2
3]** (Vector columna)
- Progresivos
 - » **0:0.1:10**
 - » **linspace(0,1,11)**
- Normas
 - » **norm(v,2)**
 - » **norm(v,1)**
 - » **norm(v,inf)**
- Dimensión
 - » **length(v)**



Operaciones con vectores

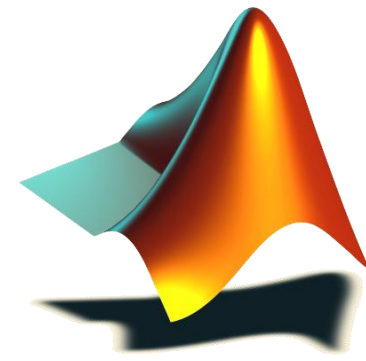
- Suma: $\mathbf{u} + \mathbf{v}$
 - de comps.: $\text{sum}(\mathbf{u})$
- Productos:
 - por escalar: $2 * \mathbf{u}$
 - escalar: $\text{dot}(\mathbf{u}, \mathbf{v})$
 - elemental: $\mathbf{u} * \mathbf{v}$
 - matricial: $\mathbf{u} * \mathbf{w}, \mathbf{w} * \mathbf{u}$
 - de comps.: $\text{prod}(\mathbf{u})$
- Transpuesta: \mathbf{u}'



Matrices y vectores

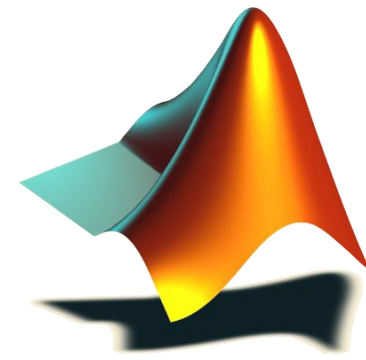
- Cómo extraer o modificar partes de matrices.
 - Podemos acceder a los elementos:
 - Vectores: Se accede a los elementos poniendo el índice entre paréntesis
 - Matrices: Se acceden poniendo los dos índices entre paréntesis o con un solo subíndice
 - Podemos modificar los elementos:
 - Asignando un valor a elementos de la matriz
 - Podemos extraer submatrices de la matriz
 - Usando el operador “:”

Ejercicio 1



Crear una matriz D , de dimensión 10×10 con todas sus componentes nulas excepto la primera fila y la primera columna, las cuales toman respectivamente los valores de la columna o fila en la que se encuentren.

Ejercicio 1



Crea una matriz D, de dimensión 10x10 con todas sus componentes nulas excepto la primera fila y la primera columna, las cuales toman respectivamente los valores de la columna o fila en la que se encuentren.

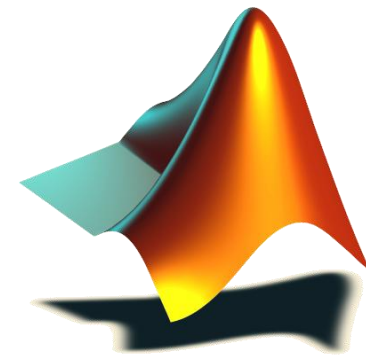
$D(1:10,1)=[1:10]'$

crea un vector columna

$D(1, 1:10)=[1:10]$

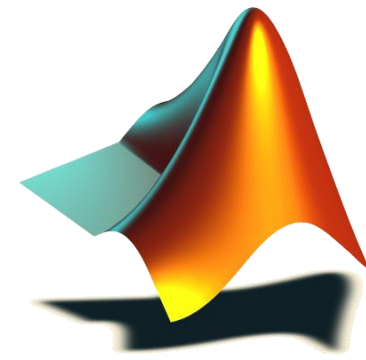
añade los ceros necesarios para construir la matriz

Ejercicio 1



Ahora modifica los elementos nulos por los elementos de valor -3.

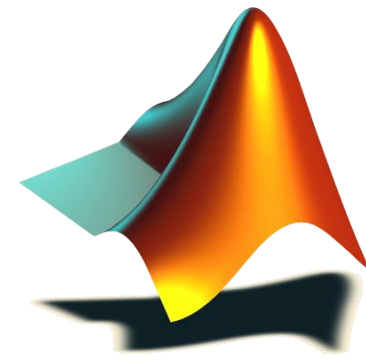
Ejercicio 1



Ahora modifica los elementos nulos por los elementos de valor -3.

$$D(2:10,2:10)=-3*\text{ones}(9,9)$$

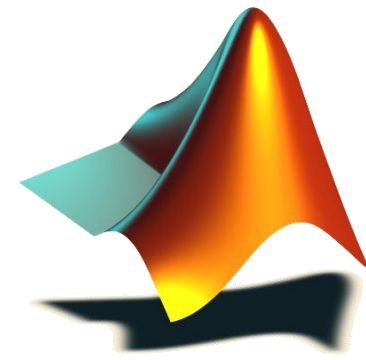
Ejercicio 2



Construye la siguiente matriz usando el operador “;” y matrices usuales (sin introducir los datos elemento a elemento).

» $A = [1 \ 1 \ 1 \ 1 \ 1 ; 1 \ 0 \ 0 \ 0 \ 0 ; 0 \ 2 \ 0 \ 0 \ 0 ; 0 \ 0 \ 3 \ 0 \ 0];$

Ejercicio 2



Construye la siguiente matriz usando el operador “;” y matrices usuales (sin introducir los datos elemento a elemento).

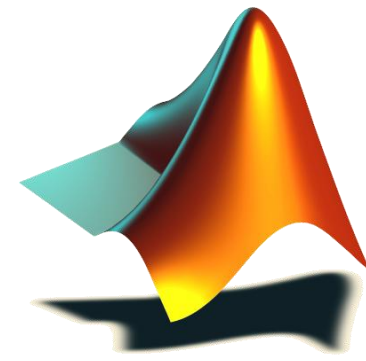
» **A=[1 1 1 1 1 ; 1 0 0 0 0; 0 2 0 0 0; 0 0 3 0 0];**

F=zeros(4,5)

F(1,:)=1

F(2:4,1:3)=diag([1 2 3])

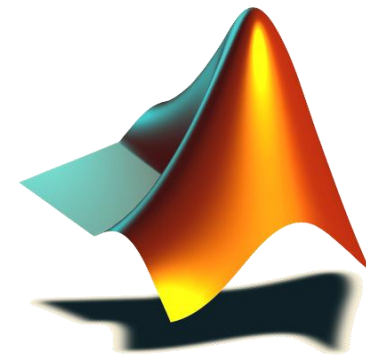
Ejercicio 3



Obtener de cuatro maneras distintas la submatriz formada por la segunda y la tercera fila de la siguiente matriz.

» $A = [1,1,1,1; 1,2,2,2; 1,2,3,3; 1,2,3,4];$

Ejercicio 3



Obtener de cuatro maneras distintas la submatriz formada por la segunda y la tercera fila de la siguiente matriz.

» `A=[1,1,1,1;1,2,2,2;1,2,3,3;1,2,3,4];`

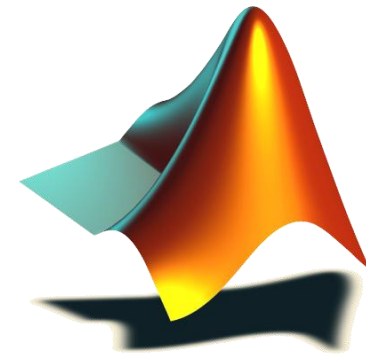
» `A(2:3,1:4)`

» `A(2:3,:)`

» `A([2 3],[1 2 3 4])`

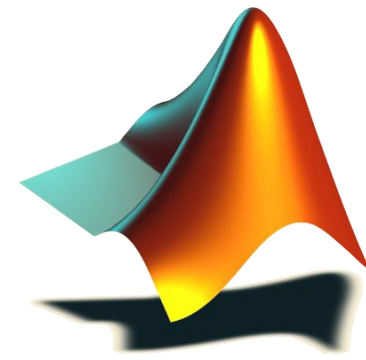
» `A([1 4],:)=[]`

Ejercicio 4



Definir una función *dettra* que tome dos matrices de entrada A y B y devuelva una variable P con el producto de las matrices y un vector v con el determinante y la traza del producto de ambas.

Ejercicio 4

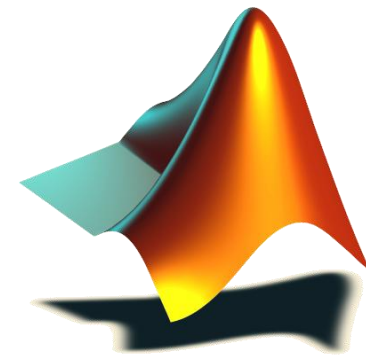


Definir una función *dettra* que tome dos matrices de entrada A y B y devuelva una variable P con el producto de las matrices y un vector v con el determinante y la traza del producto de ambas.

Crear un fichero en matlab *dettra.m*

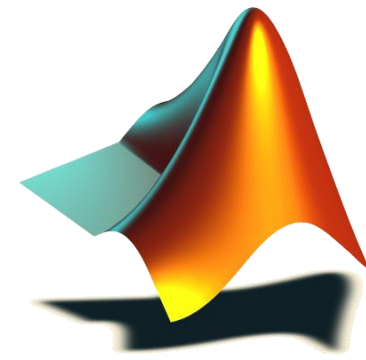
```
function [P,v]=dettra(A,B)
P=A*B;
v=[det(P), trace(P)]
```

Ejercicio 5



Escriba y guarde en un archivo de nombre `datos.m` la matriz cuadrada de orden veinte tal que los elementos de su diagonal son todos iguales a 3 y las dos subdiagonales principales están formadas por unos. Calcule su determinante.

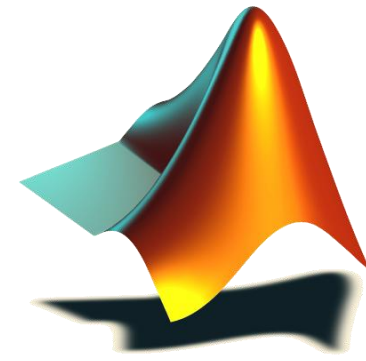
Ejercicio 5



Escriba y guarde en un archivo de nombre datos.m la matriz cuadrada de orden veinte tal que los elementos de su diagonal son todos iguales a 3 y las dos subdiagonales principales están formadas por unos. Calcule su determinante. Escribir en un archivo:

```
A=diag(3*ones(20,1));  
A=A+diag(ones(19,1),1);  
A=A+diag(ones(19,1),-1);  
Guardar en datos.m
```

**Ejecutar el archivo y
escribir en MATLAB:**
>>det(A)



Gráficas en MATLAB

- Gráficas bidimensionales:

Admite cuatro opciones:

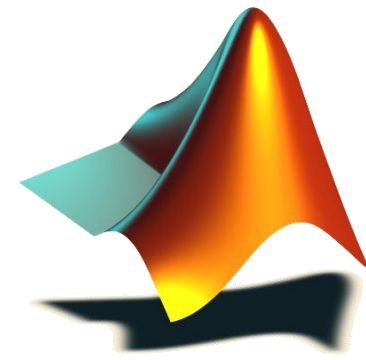
- Gráficas en coordenadas cartesianas. Usando la orden **plot(x,y)** o bien si queremos dibujar una curva que pasa por unos puntos dados (x_1, y_1) , (x_2, y_2) : **Plot([x_1,x_2],[y_1,y_2])**
- Gráficas en coordenadas polares. **polar(ángulo)**
- Gráficas de barras. **Bar()**
- Gráficas de escaleras. **Stairs()**

- Gráficas tridimensionales:

Admite tres opciones:

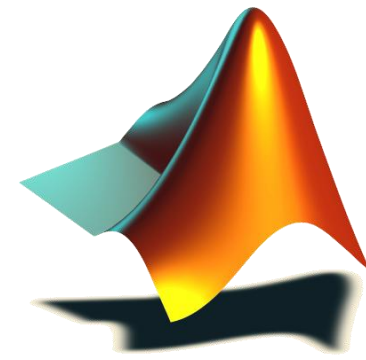
- Gráficas de líneas. Usando la la opción **plot3(x,y,z)**.
- Gráficas de superficies. **mesh** y **surf**
- Gráficas de contorno. **contour**

Ejercicio 6



Dibuje la gráfica de la función exponencial en el intervalo $[-2,2]$. Obtenga una segunda gráfica donde a la curva anterior se le añada la recta tangente en $x=0$.

Ejercicio 6



Dibuje la gráfica de la función exponencial en el intervalo $[-2,2]$. Obtenga una segunda gráfica donde a la curva anterior se le añada la recta tangente en $x=0$.

```
x=-2:0.01:2;  
y=exp(x);  
plot(x,y)
```

```
z=x+1;  
plot(x,y,x,z)
```