1 *Interface, operaciones y representación*

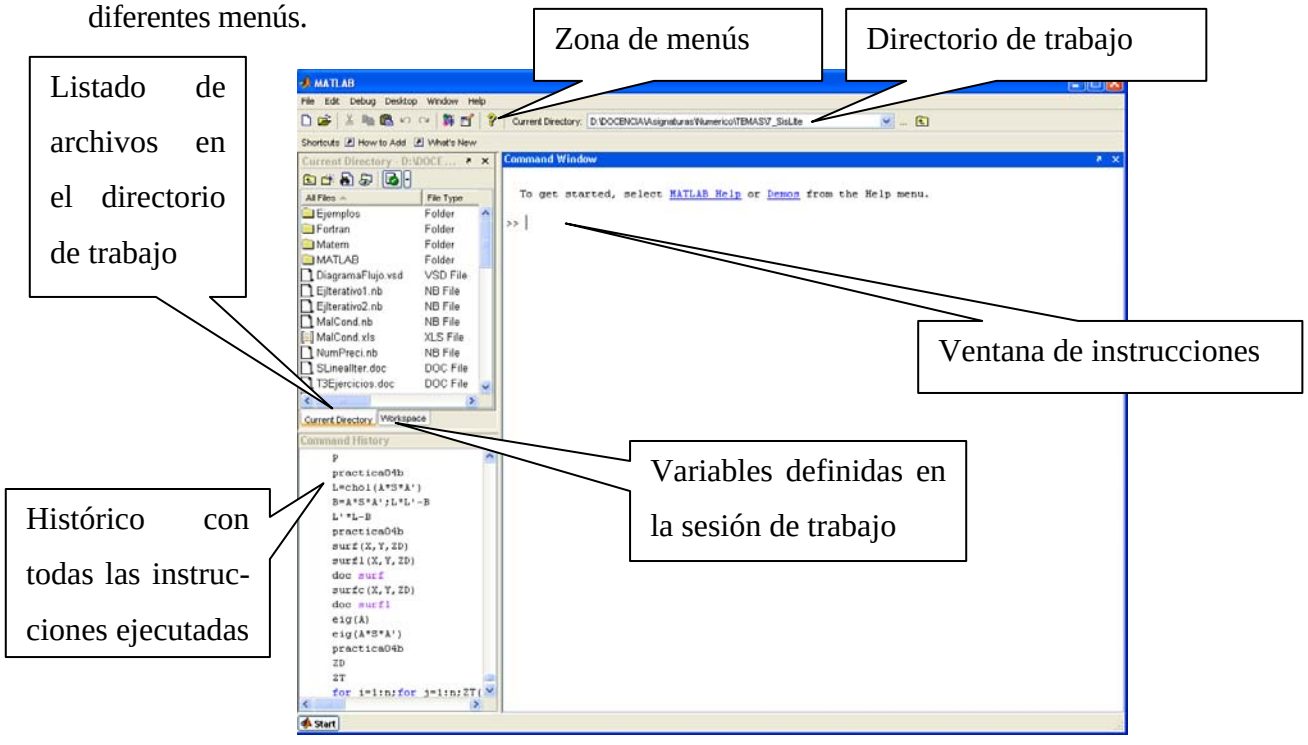
En esta práctica aprenderemos a entrar en MATLAB y utilizarlo como una potente calculadora. También veremos cómo representar pares de puntos.

1.1 Entrar en Matlab

Si aparece el icono de MATLAB como acceso directo en la pantalla inicial, es suficiente pulsar sobre el con el ratón. En otro caso, será necesario buscarlo a partir del menú de inicio.

Icono de MATLAB

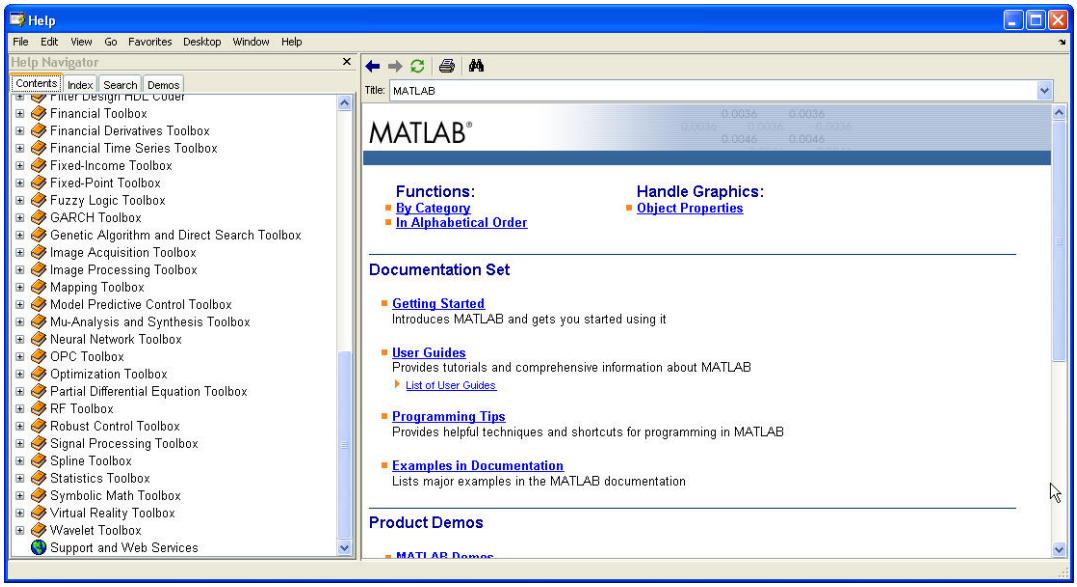
Una vez iniciado MATLAB, nos encontramos con la pantalla de la figura, donde se observan los diferentes menús.



Pantalla de entrada de MATLAB

1.2 La ayuda en MATLAB

A la ayuda se accede mediante la opción “Full Product Family Help” o “Help Window” del menú *Help*. El mismo efecto se logra con la tecla “F1”.



Pantalla de ayuda (Help Window) de MATLAB

Se puede recorrer el índice seleccionando los diferentes tópicos o bien pedir información específica sobre una función determinada utilizando las opciones “index” o “search”. Cuando se conoce el nombre de la instrucción (p.e. rand) se pueden utilizar las instrucciones **help** o **doc**.

```
» help rand
» doc rand
```

1.3 MATLAB como calculadora

Se puede usar MATLAB como si fuera una potente calculadora, almacenando variables y realizando operaciones entre ellas. MATLAB trabaja habitualmente con valores matriciales, de ahí que su definición y manejo sean fundamentales. Para definir una matriz en MATLAB, basta con introducir entre corchetes todas sus filas separadas por punto y coma. Los vectores se pueden introducir separando sus componentes por espacios en blanco o por comas. Su sintaxis es la siguiente:

vector=[a, b, c, d, …m] Define un vector fila, cuyos elementos son los valores a, b, c, d, …m.

vector=[a; b; c;, d; …m] Define un vector columna, cuyos elementos son los valores a, b, c, d, …m.

En resumen, las comas separan elementos de un vector, mientras que el punto y coma separa las filas. MATLAB indica un error cuando las filas tienen diferente número de elementos. En el apartado 4.2Selección de los elementos de un vector x o matriz se muestra la forma de manejar los elementos de este tipo de variables.

Veamos algunos ejemplos. Para generar la variable A que almacena una matriz de dos filas y tres columnas con los valores dados, introducimos

```
» A=[1,2,3 ; 4, 5, 6]
1  2  3
4  5  6
```

Para obtener la submatriz resultante de quitar la primera columna, añadir una matriz a sí misma, obtener sus todos sus elementos como un único vector o trasponerla, haremos respectivamente:

```
» A(:,2:3)
» [A A]
» A(:)
» A'
```

Existen en MATLAB dos tipos de operaciones aritméticas: Las operaciones aritméticas matriciales, que se rigen por las reglas del álgebra lineal, y las operaciones aritméticas con vectores, que se realizan elemento a elemento.

Símbolo	Operación
+	Suma de escalares, vectores o matrices
*	Producto matricial
^	Potenciación matricial
/	Cociente matricial, B/A=B*inv(A)
\	Cociente matricial, A\B=inv(A)*B

Símbolo	Operación
-	Resta de escalares, vectores o matrices
.*	Producto elemental
.^	Potenciación elemental A.^B $\rightarrow a_j^k \wedge b_j^k$
./	Cociente elemental A./B $\rightarrow a_j^k / b_j^k$
.\	Cociente elemental A.\B $\rightarrow b_j^k / a_j^k$

Hay que destacar la suma y multiplicación de un escalar por una matriz. En ambos casos se aplica la operación a cada elemento de la matriz:

```
»10*A, 10+A
ans =
    10    20    30
    40    50    60
ans =
    11    12    13
    14    15    16
```

Notemos que, como el resto de lenguajes de programación, la asignación (signo "=" en MATLAB) se puede interpretar como: "Realizar las operaciones a la derecha del símbolo = y almacenar su resultado en la variable indicada a la izquierda del =". Esto permite realizar operaciones como:

```
» A=A*A'
» i=10,i=i+2
```

Para informarnos de las variables almacenadas que estamos usando pondremos *who* o *whos*.

```
» whos
  Name        Size              Bytes  Class
  A           2x2                  32  double array
  ans         2x2                  32  double array
Grand total is 8 elements using 64 bytes
```

Al igual que la calculadora, MATLAB dispone de una librería muy completa de funciones predefinidas que se corresponden con las funciones matemáticas más utilizadas. Algunos ejemplos se muestran en el anexo. Utilizándolas, se puede calcular el máximo de todos los elementos de A o la raíz cuadrada de los mismos, como

```
» max(A(:))
» sqrt(A)
» x=10;prod(1:x) % Calcula el factorial del numero 10
» x=20;sum(1:x) % Calcula el sumatorio de numero 10
```

También dispone de instrucciones para representaciones gráficas en 2 y 3 dimensiones, de las cuales la más sencilla es ***Plot*** (ver anexo). Para representar una función en un intervalo, se generan dos vectores, uno para las abscisas y otro para las ordenadas, utilizando la orden *plot* para unir todos los pares de puntos. La representación de $y=e^x$ en [-2,2] se realiza así mediante

```
» x=linspace(-2,2);y=exp(x);plot(x,y)
```

O bien

```
» x=linspace(-2,2); plot(x,exp(x))
```

2 *Representación numérica*

La representación de valores en el ordenador no siempre es exacta. En ocasiones esto se debe a que sólo se pueden almacenar un número finito de dígitos, y el valor es irracional o periódico. En otros casos se debe a las limitaciones de la representación binaria utilizada por el ordenador. Se debe indicar que MATLAB guarda los valores con casi 16 cifras decimales, si bien sólo muestra 4 en sus resultados. Veamos como se pueden mostrar los valores de diferentes maneras

Representar $\sqrt{2}$	`» sqrt(2)`
Idem con “formato largo”	`» format long; ans`
Idem con formato racional	`» format rat; ans`
Idem con 30 cifras decimales	`» vpa(ans,30)`
Representación binaria del valor 1234	`» dec2bin(1234)`
Representación decimal del valor 1234 en base 6	`» base2dec('1234',6)`

Como resultado de los errores de representación, se pierden propiedades matemáticas básicas como la asociatividad de la suma. Utilizaremos el valor `eps`.

Si a=ε/2: 1+a-1 ¿ = ? 1-1+a

```
» a=eps/2,1+a-1, 1-1+a
```

El orden altera la suma

```
» x=rand(1,1000);y=sort(x);z=y(end:-1:1);
» vpa(sum(x),20),vpa(sum(y),20), vpa(sum(z),20)
```

Ejemplo con serie geométrica

```
» format long;
» a=0;b=-10;
» x=logspace(a,b,10000);
» sx=sum(x);
» y=x(end:-1:1); sy=sum(y);
» r=x(2)/x(1);se=(x(end)*r-x(1))/(r-1);
```

3 *Programar en MATLAB*

Al igual que en los lenguajes de alto nivel, MATLAB permite crear programas utilizando programación estructurada. Para ello cuenta con condicionales, bucles y funciones. Asimismo utiliza muchos de los recursos de la programación orientada a objetos.

3.1 Definición de funciones simples

La instrucción *inline* permite crear funciones simples desde el teclado. Su sintaxis es la siguiente:

Nombre_funcion = **inline(**'*definicion en una sola línea entre comillas simples*'**)**

Ejemplo:

```
» f=inline('sqrt(exp(x))')
» p=f(4)          % Evaluación en el punto 4
» X=linspace(-1,4);Y=f(X);plot(X,Y) % Representación en el intervalo [-1,4]
```

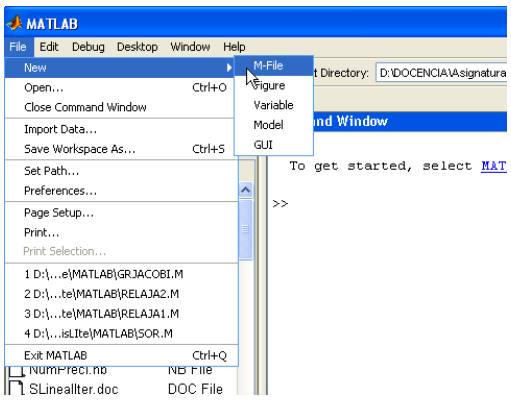
3.2 Archivos de instrucciones

Cuando se deben realizar varias instrucciones de forma consecutiva, pueden escribirse en la misma línea, separándolas por "," o ";"., o bien escribirlas en un fichero o archivo, denominado fichero de instrucciones. En el primer caso, si se utiliza "," para separarlas, el resultado es similar al de escribir cada instrucción en una línea; cuando se utiliza el ";" para separarlas, las instrucciones se ejecutan, pero no se muestran en pantalla los resultados.

```
» x=(1:5), sx=sum(x)
x=
   [1 2 3 4 5]
sx=
   15
» y=(1:10); sy=sum(y);
»sy
sy=
   55
```

Un fichero de instrucciones se puede crear utilizando el editor integrado de MATLAB, para crear un fichero de instrucciones al que se accede con la siguiente selección de menús:

"File"→"New"/"Open" → "Mfile" La selección "New/Open" depende de si vamos a crear un nuevo archivo o utilizamos un archivo creado previamente.



Edición de un archivo nuevo en MATLAB

Las instrucciones pueden ir en líneas consecutivas o bien utilizar alguno de los separadores anteriores, y se ejecutan escribiendo en MATLAB el nombre del fichero.

En ordenadores cuyo S.O. es windows NT, XP o 2003, los directorios suelen estar protegidos contra intrusiones, por lo que no es posible salvar los ficheros nuevos en cualquier lugar. En este caso suele haber un directorio "C:\alumnos" o "C:\usuarios" donde se guardan los trabajos realizados. MATLAB sólo puede ejecutar funciones que estén en memoria, en sus librerías o en el directorio actual; por ello es necesario cambiar al directorio donde salvamos nuestro archivo antes de poder ejecutarlo.

3.3 Definición general de funciones

Una función se define mediante un m-fichero, cuyo nombre coincide con el de la función. La primera línea del fichero tiene la siguiente sintaxis:

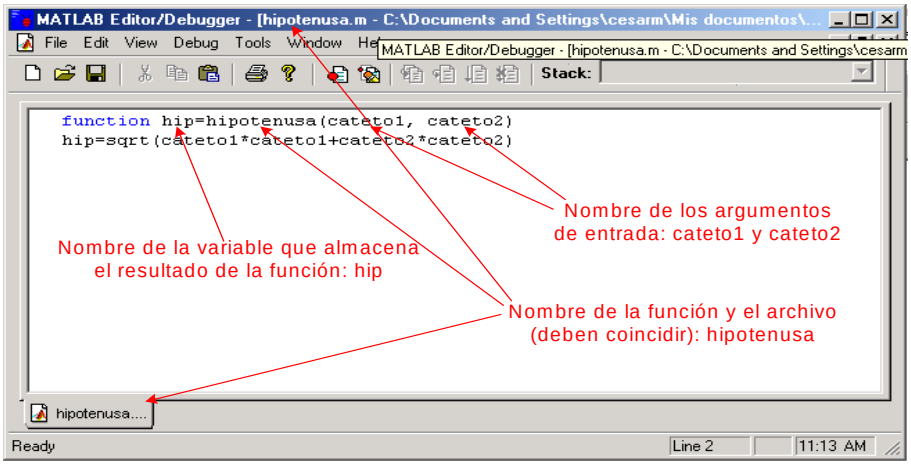
f**unction** [*argumentos_salida*]= *nombre_función (argumentos_entrada)*

Después irán el resto de instrucciones necesarias para resolver el problema planteado. Cuando hay más de un argumento (variable) de salida, éstos deben ir entre corchetes y separados por comas. Es conveniente utilizar las primeras líneas del fichero como comentarios (iniciándolas con '%'), explicando cómo debe usarse la función y sus argumentos (tanto de entrada como de salida). Así, dicha definición será visible mediante la instrucción **help** *nombre-función.*

La función puede finalizarse en cualquier punto utilizando la instrucción **return**.

Como ejemplo de lo anterior, comenzaremos creando una función que calcule el valor de la hipotenusa de un triángulo rectángulo a partir de sus dos catetos. A continuación se indican las instrucciones a programar:

```
function hip = hipotenusa(cateto1, cateto2)
hip = sqrt(cateto1*cateto1+cateto2*cateto2);
```



Función de cálculo de la hipotenusa.

Tras guardar el archivo, damos valores a dos variables y ejecutamos la nueva función desde MATLAB mediante

```
» x=3, y=4, hipotenusa(x,y)
x =
     3
y =
     4
ans =
     5
```

Observemos que la solución de la función se almacena en la variable *ans*, puesto que no se ha realizado la asignación a ninguna variable. Para recuperar el valor de una variable **existente** es suficiente con escribir su nombre (o acudir al *workspace*).

```
» x
x =
    3
```

Si se hubiera eliminado el símbolo ";" al final de la 2ª línea, habríamos obtenido el valor de la variable *hip* **durante la ejecución de la función**. Al acabar la función, dicha variable desaparece, por lo que no podremos usar su valor (no aparece en *workspace*).

```
» hip
??? Undefined function or variable 'hip'
```

Las variables definidas en la función (salvo los argumentos) son locales (no se propagan fuera del entorno de ejecución de la función). Para que el valor de una variable sea compartido por varias funciones se emplea la instrucción **global**, cuya sintaxis es **global** *variable*, y debe aparecer en todas las funciones que la compartan.

Si se quiere que las variables *x* e *y* toman ciertos valores, pero no muestran por pantalla el resultado de la asignación, acabaremos la instrucción con ";". Probemos ahora a escribir

```
» x=3; y=4;z=hipotenusa(x,y)
z =
    5
```

En resumen, se pueden separar varias instrucciones en una misma línea por una coma o un punto y coma, diferenciándose en que el segundo caso no muestra los resultados de las operaciones. De ahí que todas las instrucciones de una función acaben con punto y coma. Cuando una instrucción es demasiado larga para escribirla en una sola línea, se puede acabar con tres puntos ("`...`") indicando que continúa en la línea siguiente.

Una función predefinida por MATLAB y muy útil es **error**, cuya sintaxis es **error('Mensaje')**. Esta función finaliza la ejecución del programa actual, enviando a la pantalla un mensaje. La misma funcionalidad se puede obtener combinando las instrucciones **disp** y **return**, mediante **disp('**mensaje**');return**

Las funciones también utilizan las siguientes variables para verificar el número de argumentos:

nargin número de argumentos de entrada que el usuario ha pasado a la función.

nargout número de argumentos de salida que el usuario desea recibir de la función

Se pide:

1. Utilizando las instrucciones del anexo, representar la función y=sin(x) en el intervalo [-π,π]
2. Representar la función y=|x| en el intervalo [-2,2]

3. Representar simultáneamente las funciones sin(x),cos(x) y tan(x) en el intervalo $[-\pi,\pi]$
4. Crear los siguientes vectores
    a. Vector ordenado de pares entre 6 y 26.
    b. Vector ordenado de 10 múltiplos de tres comenzando en 21.
    c. Vector en orden inverso desde 10 hasta 0.
5. Matriz de orden 4 con todos sus elementos 3 salvo la diagonal que es nula.
6. Matriz que muestra por columnas la tabla de multiplicar desde el 1 hasta el 8
7. Función que pase un punto de coordenadas cartesianas a polares

4 *Anexo: Instrucciones necesarias*

4.1 Introducciones de manejo de vectores y matrices

Repasamos aquí las instrucciones ya introducidas cursos anteriores.

vector=[a, b, c, d, …m] Define un vector fila, cuyos elementos son los valores dados.

vector=[a; b; c;, d; …m] Idem con un un vector columna.

variable=[primer_elemento:último_elemento] Define el vector cuyos primeros y último elementos son los especificados, y los elementos intermedios se diferencian en una unidad.

variable=[primer_elemento:incremento:último_elemento] Define el vector cuyos primeros y último elementos son los especificados, y los elementos intermedios se diferencian en la cantidad especificada por el incremento

*variable=**linspace**(primer_elemento,último_elemento,n)* Define el vector cuyos primeros y último elementos son los especificados, y que tiene en total n elementos uniformemente espaciados.

*variable=**logspace**(primer_elemento,último_elemento,n)* Define el vector cuyos primeros y último elementos son los especificados, y que tiene en total n elementos en escala logarítmica uniformemente espaciados entre sí.

Para definir una matriz en Matlab, basta con introducir entre corchetes todos sus vectores fila separados por punto y coma. Los vectores se pueden introducir separando sus componentes por espacios en blanco o por comas.

4.2 Selección de los elementos de un vector x o matriz A

x(n)	Devuelve el n-ésimo elemento del vector x
x([n,m,p])	Devuelve los elementos del vector x situados en las posiciones n-ésima, m-ésima y p-ésima.
x(n:m)	Devuelve los elementos del vector x situados entre el n-ésimo y el m-ésimo, ambos inclusive
x(n:p:m)	Devuelve los elementos del vector x situados entre el n-ésimo y el m-ésimo, ambos inclusive pero separados de p en p unidades

A(m,n)	Devuelve el elemento (m,n) de la matriz A (fila m y columna n)
A([m, n],[p, q])	Devuelve la submatriz de A formada por la intersección de las filas n-ésima y m-ésima y las columnas p-ésima y q-ésima.
A(n,:)	Devuelve la fila n-ésima de la matriz A
A(:,p)	Devuelve la columna p-ésima de la matriz A
A(:)	Devuelve un vector columna cuyos elementos son las columnas de A situadas por orden
A(:,:)	Devuelve toda la matriz A
[A,B,C]	Devuelve la matriz formada por las submatrices A,B,C,...

4.3 Funciones trigonométricas e hiperbólicas

Trigonométrica	sin(Z)	cos(Z)	tan(Z)	sec(Z)	csc(Z)	cot(Z)
Trig. Inversa	asin(Z)	acos(Z)	atan(Z), atan2(Z)	asec(Z)	acsc(Z)	acot(Z)
Hiperbólica	sinh(Z)	cosh(Z)	tanh(Z)	sech(Z)	csch(Z)	coth(Z)
Hip. Inversa	asinh(Z)	acosh(Z)	atanh(Z)	asech(Z)	acsch(Z)	acoth(Z)

4.4 Funciones exponenciales

exp(Z)	Función exponencial de base e
sqrt(Z)	Función Raíz cuadrada
log(Z)	Función Logaritmo neperiano
log10(Z)	Función Logaritmo decimal

4.5 Funciones específicas de variable numérica

abs(Z)	Módulo o valor absoluto
angle(Z)	Argumento
conj(Z)	Complejo conjugado
imag(Z)	Parte imaginaria
real(Z)	Parte real
sign(x)	Signo del real x (1 si x>0, -1 si x<0)
rem(a,b)	Da el resto de la división entre los reales a y b
fix(x)	Elimina la parte decimal del real x
floor(x)	Redondea los decimales al menor entero más cercano
ceil(x)	Redondea los decimales al mayor entero más cercano
round(x)	El entero más próximo al real x

4.6 Funciones Matriciales

max(x)	Máximo de los números los elementos de x
min(x)	Mínimo de los números los elementos de x
size(A)	Filas y columnas de la matriz A
gcd(x)	Máximo común divisor de los elementos de x
lcm(x)	Mínimo común múltiplo de los elementos de x
length(x)	Longitud del vector x

4.7 Operaciones de matrices

transpose(A)	Traspuesta de la matriz A
A'	Traspuesta de la matriz A
inv(A)	Inversa de la matriz A (A^{-1})
zeros(m,n)	Crea una matriz con m filas y n columnas cuyos elementos son ceros
det(A)	Determinante de la matriz A
trace(A)	Suma de los elementos de la diagonal de A
norm(A)	Norma de A (mayor valor singular de la matriz A)
ones(m,n)	Crea una matriz con m filas y n columnas cuyos elementos son unos

eye(n)	Crea la matriz identidad de orden n
diag(v)	Crea una matriz diagonal con los elementos de v

4.8 Variable especiales

pi	3'1415926535897...
i ó j	Unidad imaginaria (raíz cuadrada de -1)
inf	Infinito, por ejemplo 1/0
eps	Menor valor positivo que sumado a la unidad tiene representación diferente a 1.
NaN	Indeterminación (Not a Number, por ejemplo 0/0)
realmin	El menor número real positivo utilizable
realmax	El mayor número real positivo utilizable
ans	Variable creada automáticamente para representar el último resultado

4.9 Gráficos bidimensionales (2-D)

plot(X)	Representa los puntos (k,X_k). Si X es una matriz, hace lo mismo para cada columna de la matriz. Si X es un vector complejo, representa Real(X) frente a IMAG(X)..
plot(X,Y)	Representa el conjunto de puntos (X,Y). Si X o Y son matrices, representa por filas o columnas los datos de X frente a los datos de Y, dependiendo si el otro vector es fila o columna. Para valores complejos de X e Y, se ignoran las partes imaginaria.
grid	Sitúa rejillas en los ejes de un gráfico 2-D o ·3-D. La opción grid on coloca las rejillas y greed off las elimina. La opción grid permuta entre on y off
hold	Permite mantener el gráfico existente con todas sus propiedades, de modo que el siguiente gráfico que se realice se sitúe sobre los mismos ejes y se superponga al existente. La opción hold on activa la opción y hold off la elimina. La opción hold permuta entre on y off. Válido para 2-D y 3-D
subplot(m,n,p)	Divide la ventana gráfica en mxn subventanas y coloca el gráfico corriente en la ventana p-ésima, empezando a contar por la parte superior izquierda y de izquierda a derecha hasta acabar la línea, para pasar a la siguiente
plot(X,Y,S)	Gráfica de plot(X,Y) con la opciones definidas en S. Usualmente, S se compone de dos caracteres entre comillas simples, el primero de los cuales fija el color de la línea del gráfico, y el segundo fija el carácter a usar en el graficado. Los valores posibles de colores y caracteres son, respectivamente, los siguientes:

y	amarillo	g	verde	.	puntos	-	sólido
m	magenta	b	azul	o	círculos	:	a puntos
c	cyan	w	blanco	x	x-marcas	-.	guiones y puntos
r	rojo	k	negro	+	signo más	--	semisólidos
				*	estrellas		

4.10 Titulos, etiquetas, mallas y textos

title('texto')	Añade el texto como título del gráfico en la parte superior del mismo	text(x,y,'texto')	Sitúa el texto en el punto (x,y)
xlabel('texto')	Sitúa el texto al lado del eje x	ylabel('texto')	Sitúa el texto al lado del eje y
gtex('texto')	Permite situar el texto en un punto seleccionado con el ratón	ginput(n)	Permite recuperar las coordenadas de n puntos de un grafico mediante ratón