

# **TUTORIAL DE INTRODUCCIÓN A MATLAB**

**Manuel Vargas Villanueva**

Este tutorial está basado en un trabajo original de:  
**Manuel Berenguel Soria y Teodoro Álamo Cantarero**

# Contenido

<b>1</b>	<b>INTRODUCCIÓN A MATLAB</b>	<b>1</b>
1.1	INTRODUCCIÓN . . . . .	1
1.2	INSTALACIÓN . . . . .	2
1.3	PRIMEROS PASOS . . . . .	4
1.4	FUNCIONES Y SÍMBOLOS RELACIONADOS CON EL ENTORNO . . . . .	4
1.5	INTRODUCCIÓN DE DATOS. USO DE LA VENTANA DE COMANDOS . . . . .	6
1.6	VARIABLES DE ENTORNO Y VARIABLES ESPECIALES . . . . .	7
1.7	ELEMENTOS DE LAS MATRICES . . . . .	8
1.8	OPERACIONES CON MATRICES . . . . .	9
1.9	FUNCIONES ORIENTADAS AL ANÁLISIS DE DATOS . . . . .	10
1.10	POLINOMIOS . . . . .	10
1.11	OTRAS FUNCIONES DE INTERÉS . . . . .	11
1.12	GRÁFICOS . . . . .	12
1.13	PROGRAMANDO EN MATLAB . . . . .	14
1.13.1	Operadores lógicos y relacionales . . . . .	14
1.13.2	Bucles y estructuras condicionales . . . . .	14

1.13.3	Ficheros .m . . . . .	16
1.14	RESUMEN DE LOS COMANDOS DE MATLAB . . . . .	18

# Capítulo 1

## INTRODUCCIÓN A MATLAB

### 1.1 INTRODUCCIÓN

En estas notas se pretende realizar una introducción muy básica a MATLAB, orientándola en el siguiente capítulo al estudio de sistemas de control. En líneas generales, MATLAB es una herramienta interactiva basada en matrices para cálculos científicos y de ingeniería (de hecho, el término MATLAB procede de *matrix laboratory*). Desde el punto de vista del control, MATLAB se puede considerar un entorno matemático de simulación que puede utilizarse para modelar y analizar sistemas. Permitirá el estudio de sistemas continuos, discretos, lineales y no lineales, mediante descripción interna y externa, en el dominio temporal y frecuencial.

MATLAB constituye un entorno abierto, para el cual numerosas paquetes específicos adicionales (*toolboxes*) han sido desarrollados. En el caso que nos ocupa se utilizará fundamentalmente el *Control System Toolbox*. Estos paquetes específicos adicionales están constituidos por un conjunto de funciones que pueden ser llamadas desde el programa y mediante las cuales se pueden realizar multitud de operaciones.

Las referencias al *Control System Toolbox* se realizarán directamente en los ejemplos que acompañan a estas notas.

Las notas se centrarán fundamentalmente en aquellos aspectos y funciones que más interés tengan desde el punto de vista de control, instando al lector a que busque en el manual de usuario cualquier información adicional que desee ([4], [3], [2]). Para el desarrollo de las mismas se ha utilizado asimismo, una serie de referencias básicas en control: [1], [5], [6], [7], etc.

## 1.2 INSTALACIÓN

La forma normal en la que se encuentra el sistema una vez instalado es la siguiente (versión 3.5.1):

```
\matlabr11\bin
  \extern
  \help
  \notebook
  \simulink
  \sys
  \toolbox
    \control
    \local
    \matlab
    \simulink
  \work
```

El núcleo fundamental de MATLAB se encuentra en los subdirectorios **BIN** y **MATLAB**. En **BIN** se encuentran los programas ejecutables. El subdirectorio **MATLAB** contiene los ficheros **.m** (aunque serán explicados posteriormente, comentamos brevemente que consisten en ficheros escritos a base de comandos de MATLAB y que realizan una función determinada), que contienen las funciones básicas para el funcionamiento de MATLAB. En este sentido, es necesario comentar que MATLAB cuenta con dos tipos básicos de funciones:

Funciones denominadas *built-in functions*: Son funciones que MATLAB tiene incorporadas internamente y por tanto no son accesibles al usuario.

Funciones llamadas *m functions*: Son funciones cuyo código es accesible. Las que se encuentran en el subdirectorio **MATLAB** son las básicas para el funcionamiento del sistema.

Como se desprende del árbol de directorios, los *toolboxes* se suelen instalar en forma de subdirectorios en el disco duro, colgando del subdirectorio **TOOLBOX**. En ellos se encuentran también funciones **.m** orientadas al control de sistemas. Además, se pueden incorporar otros *toolboxes* (SIGNAL PROCESSING, IMAGE PROCESSING, ROBUST CONTROL, NON-LINEAR CONTROL, SYSTEM IDENTIFICATION, etc), e incluso funciones propias del usuario.

<code>matlab\general</code>	- Comandos de propósito general
<code>matlab\ops</code>	- Operadores y caracteres especiales
<code>matlab\lang</code>	- Constructores del lenguaje de programación
<code>matlab\elmat</code>	- Matrices elementales y manipulación matricial
<code>matlab\elfun</code>	- Funciones matemáticas elementales
<code>matlab\specfun</code>	- Funciones matemáticas especiales
<code>matlab\matfun</code>	- Funciones matriciales - álgebra lineal numérica
<code>matlab\datafun</code>	- Análisis de datos y transformada de Fourier
<code>matlab\polyfun</code>	- Interpolación y polinomios
<code>matlab\funfun</code>	- Funciones de funciones y métodos para ODE
<code>matlab\sparfun</code>	- Funciones para matrices dispersas
<code>matlab\graph2d</code>	- Gráficos en dos dimensiones
<code>matlab\graph3d</code>	- Gráficos en tres dimensiones
<code>matlab\specgraph</code>	- Gráficos especializados
<code>matlab\graphics</code>	- Manipulación de gráficos
<code>matlab\uitools</code>	- Herramientas de interfaz gráfica de usuario (GUI)
<code>matlab\strfun</code>	- Cadenas de caracteres
<code>matlab\iofun</code>	- Funciones para entrada/salida de ficheros
<code>matlab\timefun</code>	- Hora y fecha
<code>matlab\datatypes</code>	- Tipos de datos y estructuras
<code>matlab\winfun</code>	- Ficheros de interfaz con Windows (DDE/ActiveX)
<code>matlab\demos</code>	- Ejemplos y demostraciones
<code>simulink\simulink</code>	- Simulink
<code>simulink\blocks</code>	- Librería de bloques de Simulink
<code>simulink\simdemos</code>	- Ejemplos y demostraciones de Simulink
<code>toolbox\control</code>	- Paquete de Control de Sistemas
<code>toolbox\local</code>	- Librería de funciones locales

Tabla 1.1: Listado del comando `help`

### 1.3 PRIMEROS PASOS

Una vez arrancado MATLAB, se abre la ventana de comandos en la que aparece el *prompt* o línea de comandos (representado con el símbolo  $\gg$ ). Este es el momento de comentar la existencia del comando más famoso de cualquier aplicación: **help**. Introduciendo este comando aparecerán todas las citadas *built-in functions*, tanto las contenidas en el subdirectorio **MATLAB**, como otras contenidas en subdirectorios eventualmente añadidos por el usuario (ver Tabla 1.1).

Para obtener información sobre cualquiera de las funciones se introduce **help nombre-función**.

**Ejemplo:** *help impulse* (*impulse* es una función que calcula la respuesta impulsional de un sistema y que se encuentra en el CONTROL SYSTEM TOOLBOX).

Una cuestión importante a tener en cuenta es que MATLAB distingue entre mayúsculas y minúsculas. En este sentido, los nombres de función se introducirán en minúsculas.

El comando **demo** permite obtener una demostración de las "posibilidades" de MATLAB.

### 1.4 FUNCIONES Y SÍMBOLOS RELACIONADOS CON EL ENTORNO

- Con el comando **path** puede comprobarse cuáles son las localizaciones de los ficheros y programas con los que va a trabajar MATLAB, pudiendo añadirse nuevos subdirectorios (incluso personales) a conveniencia. La forma más cómoda de interactuar con dichas localizaciones es mediante la opción *File/Set-Path...* en el menú de la ventana de comandos. Para poder usar cualquier función **.m**, como por ejemplo las contenidas en el paquete de control, bastará con que el camino `\matlabr11\toolbox\control` esté incluido en el *path* de MATLAB (cosa que ocurrirá si el paquete se instaló adecuadamente).
- Por otro lado, MATLAB comienza trabajando, por defecto, en el subdirectorio **matlabr11\work**. Si queremos cambiar de *directorio de trabajo* en cualquier momento, podemos hacerlo con el comando **cd camino**. Puede utilizarse en nombre completo del comando si se desea: **chdir**. Cabe decir que todas las funciones **.m** que existan en el directorio de trabajo serán localizadas sin necesidad de tener que incluir dicho directorio en el *path* de MATLAB.
- El comando **pwd** nos indica cuál es el directorio de trabajo actual.
- Para mostrar el contenido del directorio de trabajo, se pueden emplear los comandos **dir** ó **ls**. El comando **delete nombre-fichero** puede emplearse para eliminar un archivo del directorio de trabajo. Asimismo, se pueden realizar operaciones típicas de línea de comandos del sistema operativo DOS, introduciendo el comando correspondiente precedido por el símbolo "!".

- Resulta interesante tener en cuenta que la línea de comandos de MATLAB posee "memoria" y podemos recuperar comandos introducidos previamente, haciendo uso de las teclas de movimiento de cursor arriba y abajo. Para una localización más eficaz de algún comando introducido previamente, podemos teclear los primeros caracteres del mismo antes de usar el cursor arriba y sólo buscará entre los comandos ya introducidos aquéllos cuyos primeros caracteres coincidan con los introducidos.
- Otra posibilidad que se ofrece es la de introducir varios comandos en una misma línea de la ventana de comandos, separados por coma o punto y coma.
- Puede "limpiarse" el contenido de la ventana de comandos mediante la instrucción `clc`.
- El símbolo `%` sirve para introducir comentarios. Todo lo escrito desde ese símbolo hasta el final de la línea será ignorado por el intérprete de MATLAB. El uso de comentarios puede no resultar demasiado interesante en la línea de comandos, aunque sí lo será cuando se estén escribiendo programas, como se verá más adelante.
- Si se quiere guardar toda la sesión en un archivo, basta usar el comando `diary nombre-archivo`. Dicho archivo contendrá los comandos introducidos y los correspondientes resultados. Cuando no se quiera seguir almacenando la información se introducirá `diary off`.
- Si se desean almacenar todas las variables de memoria en un fichero, junto con sus valores actuales, se usa el comando `save nombre-fichero`. Esto crea un fichero binario en el directorio de trabajo actual con el nombre introducido y con extensión `.mat`. Si no se da el nombre del fichero, se crea uno llamado `matlab.mat`. En caso que se desee guardar en un fichero con formato ASCII, se introducirá en el comando un modificador: `save -ascii nombre fichero`. Si sólo se quieren guardar una serie de variables, se introducirá `save nombre-fichero nombre-variables` separadas por espacios.
- Para recuperar los ficheros generados con el comando `save` se utilizará `load nombre-fichero`.
- El formato de visualización en la ventana de comandos puede modificarse usando `format`:
  - `format long`: Presentará mayor número de decimales en pantalla al presentar los resultados en punto flotante.
  - `format short`: Es el modo por defecto, presenta un número de decimales menor. Este formato no afecta para nada a la precisión de los cálculos, es sencillamente una cuestión de visualización.
  - `format compact`: Deja menor número de líneas en blanco en la visualización de los resultados, permitiendo dar cabida a más información previa en la ventana de comandos sin necesidad de hacer *scroll*.
  - `format loose`: Es el modo por defecto, se dejan más líneas de separación durante la visualización.

También puede modificarse el formato de visualización a través de las opciones de menú: *File/Preferences/General*

- Para detener la ejecución de un comando, se usa `Ctrl-C`.
- La salida del sistema se efectúa al introducir `quit` ó `exit`, o simplemente cerrando la ventana de comandos.

## 1.5 INTRODUCCIÓN DE DATOS. USO DE LA VENTANA DE COMANDOS

El elemento básico en MATLAB es **la matriz compleja de doble precisión**, estructura que abarca realmente todo tipo de datos, desde escalares tales como números reales o complejos, hasta vectores o matrices de tamaños arbitrarios. Implícitamente se usa la notación matricial para introducir polinomios y funciones de transferencia, de la forma que se explicará más adelante. Por otro lado, si se dispone de una representación de un sistema lineal en el espacio de estados de la forma:

$$\begin{aligned}\dot{x} &= A x + B u \\ y &= C x + D u\end{aligned}$$

bastaría con introducir los valores de los elementos de las matrices  $A$ ,  $B$ ,  $C$  y  $D$ , para tener descrito al sistema. Estos elementos se podrían introducir de la siguiente forma:

```
A=[1 0 2;2 2 0;0 0 1]
B=[1, 0,0]'
C=[1 1 sqrt(2)]
D=0;
```

A la vista de esta serie de comandos se pueden comentar varias cosas:

- Si al final de la introducción de un comando cualquiera no se pone punto y coma (;), aparecerá explícitamente en pantalla el resultado de dicho comando. En caso contrario, el comando se ejecutará pero no se mostrará su resultado. Dicho resultado se habrá almacenado en la variable a la que se asigna o, si no se realiza asignación, se guardará en una variable de entorno llamada `ans`. En caso de que se asigne a una variable, ésta se creará automáticamente, sin necesidad de una declaración previa.
- Los elementos de cada fila de una matriz se pueden introducir separados por espacios o por comas, indistintamente.
- Para separar filas de una matriz se usa ; o un simple retorno de carro. Esta última opción puede facilitar muchas veces la visualización de la matriz que se está introduciendo.
- Para transponer matrices se usa el apóstrofe.
- Los elementos de vectores y matrices pueden ser reales, complejos e incluso expresiones, como vemos en el caso del último elemento del vector  $C$ .
- Si se está introduciendo un comando o conjunto de ellos cuya sintaxis sea muy larga, se puede continuar en la siguiente línea introduciendo al final de la actual tres puntos seguidos (...).
- Las variables a las que se asignan resultados, así como las variables de entorno, se almacenan en lo que se denomina *el espacio de trabajo* de MATLAB (*workspace*).

En este caso, se han creado una serie de variables (en particular, matrices) mediante la introducción explícita de sus elementos en línea de comandos. Otras formas de producir variables podrían ser: generándolas mediante funciones y declaraciones, creándolas en un archivo `.m`, cargándolas desde un archivo de datos externo mediante el comando `load` (bien se trate de ficheros de datos ASCII o bien de ficheros binarios con formato de datos de MATLAB `.mat`).

Además de variables numéricas, escalares o matriciales, en MATLAB pueden usarse cadenas de caracteres. Para ello se delimita una secuencia de caracteres mediante apóstrofes:

```
cadena = 'ejemplo de cadena de caracteres'
```

Para hacer referencia a cualquiera de los caracteres que componen una cadena, podemos hacerlo como si de un vector se tratara (la forma de indexar vectores y matrices se verá más adelante).

## 1.6 VARIABLES DE ENTORNO Y VARIABLES ESPECIALES

Existen una serie de variables predefinidas en MATLAB, son las siguientes:

- `ans`: Contiene la respuesta (*answer*) del último comando ejecutado, cuando el resultado de dicho comando no se asigna explícitamente a ninguna variable.
- `eps`: Da el valor de la precisión con la que la máquina realiza las operaciones en punto flotante. Típicamente, esta precisión es del orden de  $10^{-17}$ .
- `pi`:  $\pi$ .
- `i`, `j`:  $\sqrt{-1}$ . Constante imaginaria.
- `inf`:  $\infty$ . Se trata de un valor excesivamente grande para ser almacenado.
- `NaN`: *Not a number*. Es el resultado que se proporciona si durante una operación se produce una indeterminación, del tipo  $0 \cdot \infty$ ,  $\frac{0}{0}$ ,  $\frac{\infty}{\infty}$ , etc.
- `clock`: Reloj.
- `date`: Fecha.
- `flops`: Número de operaciones en punto flotante realizadas hasta el momento.

El comando `who` muestra las variables existentes en el espacio de trabajo generadas por el usuario, pero no las variables especiales.

Para borrar alguna variable de memoria se utiliza `clear nombre-variables` separadas por espacios. Pueden borrarse todas las variables a la vez si no se especifica ningún nombre a continuación del nombre del comando.

## 1.7 ELEMENTOS DE LAS MATRICES

En este punto es importante comentar uno de los elementos más potentes de MATLAB, que es el símbolo `:`, que permite generar una secuencia, y en particular permitirá referenciar varios elementos de una matriz. Veamos algunos ejemplos en los que se usa este operador:

`1:0.1:10` Generará una secuencia comenzando por 1 hasta 10, cada elemento de la secuencia estará separado del anterior en 0.1.

`1:10` Si se obvia el valor central, la separación entre cada dos elementos de la secuencia será 1.

`[1:0.1:10]` Si lo ponemos entre corchetes, estaremos generando un vector con los elementos de la secuencia.

En la forma más directa, los elementos de una matriz se referencian mediante  $A(i, j)$ , donde  $i$  y  $j$  son los índices del elemento correspondiente. Podemos usar una secuencia que facilitar la indexación de múltiples elementos, como en los siguientes ejemplos:

`A(1,2:3)` daría como resultado los elementos de las columnas 2 y 3 pertenecientes a la primera fila.

`A(:,2)` daría como resultado todos los elementos pertenecientes a la segunda columna.

Lógicamente, en estos casos, los elementos especificados como inicio, final e incremento para producir la secuencia deben ser enteros.

Otra forma de generar datos secuencialmente es usando los comandos `linspace` y `logspace`, su formato es:

```
t = linspace(n1,n2,n);  
w = logspace(n1,n2,n);
```

El comando `linspace` genera un vector desde  $n1$  a  $n2$  de longitud  $n$ , cuyos componentes poseen valores espaciados linealmente. Por su parte, `logspace` produce también un vector de  $n$  elementos, pero sus valores están espaciados logarítmicamente desde  $10^{n1}$  a  $10^{n2}$ . Este último comando resultará útil para la generación de escalas frecuenciales para el análisis de sistemas mediante diagramas de Bode, Nyquist, etc.

## 1.8 OPERACIONES CON MATRICES

Las operaciones comunes con matrices son:

- Suma: `+`
- Resta: `-`
- Multiplicación: `*`
- División derecha `/` ( $x = b/A$  es la solución del sistema de ecuaciones  $x * A = b$ . Es decir calcula la inversa de la matriz  $A$  y multiplica  $b$  por la derecha por dicha inversa)
- División izquierda `\` ( $x = A \backslash b$  es la solución de  $A * x = b$ . Es decir, igual que en el caso anterior, pero realiza la multiplicación de la inversa con  $b$  por la izquierda)
- Potenciación `^`. Este operador permite, en particular, implementar otra forma de realizar la inversión de una matriz:  $A^{-1}$ .
- Conjugada traspuesta `'`

Cabe mencionar la potencia de los operadores `/`, `\`, y `^`, puesto que si la matriz  $A$  no es cuadrada, automáticamente se realiza el cálculo de su pseudoinversa, lo que equivaldría a resolver el sistema de ecuaciones correspondiente por mínimos cuadrados.

Las mismas operaciones que se han enumerado se pueden realizar elemento a elemento, anteponiendo un punto a cualquiera de los operandos anteriores. Como ejemplo, el siguiente comando realizaría el producto de cada elemento de la matriz  $A$  con su correspondiente de la matriz  $B$  (para que dicho producto sea realizable, obviamente, dichas matrices deben tener las mismas dimensiones):

```
A .* B
```

Además de los operadores anteriores, existen funciones tales como:

- Trigonómicas estándar: `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`
- Trigonómicas hiperbólicas: `sinh`, `cosh`, `tanh`, `asinh`, `acosh`, `atanh`
- Trascendentales: `log`, `log10`, `exp`, `sqrt`
- Manipulación de números complejos:
  - `real`: parte real de un escalar o de los elementos de una matriz.
  - `imag`: parte imaginaria.
  - `conj`: proporciona el conjugado de un escalar o la matriz conjugada a una dada.

- Cálculo del módulo: `abs` permite calcular tanto el valor absoluto de un escalar real como el módulo de un escalar complejo o el módulo de un vector.
- Funciones típicas de matrices:
  - `det`: determinante de una matriz
  - `inv`, `pinv`: inversa y pseudoinversa
  - `eig`: obtención de autovalores
  - `rank`: rango de la matriz
  - `norm`: norma de una matriz (norma 2, norma 1, norma infinito, norma de Frobenius)
  - `trace`: traza de la matriz
  - `diag`: produce un vector conteniendo los elementos de la diagonal de una matriz, o si recibe un vector como parámetro, genera una matriz diagonal.
  - `tril`: devuelve la matriz triangular inferior de una matriz dada
  - `triu`: devuelve la matriz triangular superior de una matriz dada
- funciones para generar matrices:
  - `eye(n)`: produce una matriz identidad de dimensión  $n \times n$
  - `zeros(n,m)`: genera una matriz de ceros de dimensión  $n \times m$
  - `ones(n,m)`: genera una matriz de unos de dimensión  $n \times m$
  - `rand(n,m)`: permite generar una matriz de valores aleatorios, entre 0 y 1, de dimensión  $n \times m$
  - `A = [A11,A12;A21,A22]`: podemos producir una nueva matriz por bloques, mediante su composición a partir de submatrices ya existentes.

## 1.9 FUNCIONES ORIENTADAS AL ANÁLISIS DE DATOS

Se trata de funciones que operan con vectores. Si se aplican a matrices operan columna a columna. Permiten realizar análisis sobre el conjunto de datos contenido en los vectores correspondientes, tales como calcular su valor mínimo, máximo, media, mediana, desviación típica, suma de los elementos de dicho vector, etc. `min`, `max`, `mean`, `median`, `std`, `sum`, `prod`, etc.

## 1.10 POLINOMIOS

Esta es una sección importante, dado que las funciones de transferencia de los sistemas se introducirán habitualmente en la forma *numerador-denominador*, los cuales serán tratados

como polinomios por MATLAB. En las demos que acompañan a estas notas se podrán analizar numerosos ejemplos.

Los polinomios se representan por vectores, cuyos elementos son los coeficientes del polinomio en orden descendente. Por ejemplo, el polinomio  $s^3 + 2s^2 + 3s + 4$  se representa:

```
p=[1 2 3 4];
```

que muy bien podría ser el denominador de una función de transferencia.

Mediante la función `roots` se pueden encontrar las raíces de esa ecuación:

```
roots(p)
```

De modo complementario, se puede calcular un polinomio a partir de sus raíces usando la función `poly`:

```
p2=poly([-1 -2]);
```

Si el argumento de entrada a `poly` es una matriz, devuelve el polinomio característico de la matriz ( $\det(\lambda I - A)$ ) como un vector fila.

Un polinomio puede ser evaluado en un punto determinado usando `polyval(p,s)`, donde  $p$  es el polinomio y  $s$  es el punto donde va a ser evaluado. Por ejemplo:

```
p2=[1 3 2]; a=[1 2; 3 4]; polyval(p2,a)
```

si se introduce, como en este caso, un vector o una matriz, en lugar de un valor individual, la evaluación se hace elemento a elemento.

Podemos realizar cómodamente operaciones de multiplicación y división de polinomios mediante las funciones `conv` y `deconv`, respectivamente:

```
conv([1,2],[2,0])
```

## 1.11 OTRAS FUNCIONES DE INTERÉS

En esta sección simplemente comentaremos, de forma rápida, la existencia de una serie de funciones muy útiles en problemas de integración numérica (`quad`, `quad8`), solución de ecuaciones diferenciales (`ode23`, `ode45` y muchos otros), importantes cuando se estudian los sistemas dinámicos, ecuaciones no lineales (`fmin`, `fsolve`, etc.), interpolación (`spline`, etc.)...

## 1.12 GRÁFICOS

MATLAB es muy potente a la hora de generar gráficos (sobre todo en sus últimas versiones), no sólo por la variedad de comandos que ofrece para ello, sino también por la versatilidad de dichos comandos. En las demostraciones aparecerán varios tipos de gráficos. De momento, comentaremos los comandos fundamentales para la realización de los mismos. En primer lugar, comandos genéricos y comandos orientados a gráficos bidimensionales:

- **figure(n)**: Las representaciones de gráficos en MATLAB se realizan en ventanas gráficas. En un momento dado puede haber varias ventanas gráficas abiertas. La función **figure** se utiliza para abrir una nueva ventana gráfica que será numerada de acuerdo con el parámetro, o bien, si ya existe una ventana con ese número, se convertirá en la ventana gráfica activa, donde se realizará la próxima representación gráfica.
- **clf**: Limpia la ventana gráfica activa.
- **close(n)**: Para cerrar una ventana gráfica. **close all** cierra todas las ventanas gráficas.
- **plot**: es la función básica de representación gráfica de datos en dos dimensiones. La representación se realiza en la ventana gráfica que esté activa en un momento dado. En caso de no haber ninguna, se crea una ventana gráfica nueva. Ejemplos de uso:
  - **plot(v)**: representa en el eje vertical los valores contenidos en el vector  $v$ , frente a los valores del índice en el eje horizontal.
  - **plot(t,v)**: representa los valores del vector  $v$  frente a los del vector  $t$ .
  - **plot(t,A)**, **plot(t,[v1,v2])**: presentará varias gráficas, puesto que cada columna de la matriz  $A$  es considerada como un vector a representar frente al vector  $t$ . En la segunda variante indicada, se consigue lo mismo mediante la agrupación de los vectores  $v1$ ,  $v2$  en una matriz.
  - **plot(t1,v1,t2,v2)**: En este caso también se obtendrán dos gráficas, pero cada una de ellas tiene un conjunto de valores diferente para el eje horizontal.
- **loglog**: representación en escala logarítmica en ambos ejes.
- **semilogx**: representación en escala semilogarítmica, el eje vertical aparecerá en escala lineal.
- **semilogy**: representación en escala semilogarítmica, el eje horizontal aparecerá en escala lineal.
- **polar**: representación de datos dados en forma polar, es decir en lugar de dar un par de vectores de componentes horizontales y verticales, se dan los vectores conteniendo el vector de ángulo y módulo.

Cuando se representan varias curvas simultáneamente en una misma ventana gráfica, se utiliza una secuencia predefinida de colores para aplicar uno diferente a cada una de ellas. Se puede

cambiar manualmente el color que por defecto tendrá una determinada curva con la adición de un parámetro: `plot(t,y,'r')`. En este ejemplo, en lugar de representarse la curva con el color por defecto (azul), aparecerá en color rojo. Para ver los códigos de colores, puede consultarse la ayuda del comando `plot`.

También pueden realizarse gráficos en tres dimensiones:

- `plot3(x,y,z)`: comando análogo a `plot` para dibujar curvas, pero en tres dimensiones.
- `mesh(x,y,Z)`: para dibujar superficies,  $Z$  debe ser una matriz con tantas filas como longitud del vector  $x$  y tantas columnas como la longitud del vector  $y$ . Los puntos que se representan son:  $(x(i), y(j), Z(i, j))$ .
- `contour`: representa en un plano horizontal las curvas de nivel de una superficie tridimensional.

Por otro lado, existen comandos que permiten añadir determinados complementos a estos gráficos:

- `title`: permite añadir un título a la gráfica
- `xlabel`: añadir una etiqueta al eje horizontal de la gráfica
- `ylabel`: añadir etiqueta al eje vertical
- `grid`: añadir una rejilla
- `axis`: permite modificar los límites de los ejes horizontal y vertical
- `text`: añadir un texto en una posición cualquiera de la gráfica
- `gtext`: igual que `text` pero permite seleccionar la ubicación del texto mediante el ratón.

Por otra parte, muchos de los elementos gráficos pueden manipularse como objetos que tienen una serie de propiedades asociadas. Por ejemplo:

```
handlePlot = plot(x,y);
```

con este comando estamos asignando el objeto de tipo `plot` a una variable. Podemos ver las propiedades asociadas a un objeto mediante la función `get(handlePlot)`, o bien especificar alguna de ellas: `get(handlePlot, 'LineStyle')`. Cualquiera de las propiedades de un objeto pueden ser alteradas mediante la función `set(handlePlot, 'Color', 'g')`.

Por otra parte, también se dispone de cierta capacidad de modificación de las gráficas mediante opciones de la propia ventana gráfica, en lugar de usar instrucciones desde la ventana de comandos.

## 1.13 PROGRAMANDO EN MATLAB

MATLAB permite a la hora de programar una serie de elementos típicos para la modificación del flujo de una secuencia de instrucciones. La sintaxis es muy parecida a la de cualquier lenguaje de programación. Todos estos operadores se pueden usar en la ventana de comandos, en línea, o en un fichero `.m`.

### 1.13.1 Operadores lógicos y relacionales

Permiten la comparación de escalares (o de matrices elemento a elemento). Si el resultado de la comparación es verdadero, devuelven un 1, en caso contrario devuelven un 0.

Los operadores elementales son:

<code>&lt;</code>	menor que	<code>&lt;=</code>	menor o igual	<code>==</code>	igual
<code>&gt;</code>	mayor que	<code>&gt;=</code>	mayor o igual	<code>~=</code>	no igual

Es importante no dejar espacios entre los operadores formados por dos símbolos. Si los datos a comparar son matrices, la comparación se hace elemento a elemento, devolviendo una matriz binaria.

### 1.13.2 Bucles y estructuras condicionales

En esta sección se explica una serie de comandos importantes a la hora de hacer un programa en MATLAB: `for`, `while`, `if-else`.

- `for`

La sintaxis de este comando es la siguiente:

```
for variable = expresion
    hacer algo;
end
```

La *expresion* es un vector, una matriz o cualquier comando de MATLAB que produzca como salida un vector o una matriz. La ejecución se realiza una vez por cada elemento del vector o de una columna de la matriz. Tanto los bucles como las estructuras condicionales se terminan con `end`.

Presentamos un primer ejemplo en el que la variable *i* toma los valores 10, 9, ..., 1:

```

for i=10:-1:1
    kk(11-i)=i;
end

```

A continuación otro ejemplo en el que aparecen dos bucles anidados:

```

x = [0:0.1:pi]';
y = x;
for f=1:length(x)
    for c=1:length(y)
        Z(f,c) = sin(x(f)).^2 + cos(y(c)).^2;
    end
end
mesh(x,y,Z);

```

Es importante evitar en lo posible el uso de bucles en MATLAB, ya que consumen mucho tiempo, pudiéndose en muchos casos realizar las mismas operaciones de una forma más eficiente y compacta.

Los siguientes ejemplos calculan logaritmos de números desde 1 a 10.000. Se hará de diferentes maneras para comparar. Se utilizan los comandos `clock` (que devuelve la hora actual) y `etime` (que devuelve el tiempo en segundos que ha transcurrido entre dos instantes) para calcular el tiempo consumido en las operaciones.

```
t1=clock; for i=1:10000, a(i)=log(i); end; e1=etime(clock,t1);
```

```
t1=clock; ind=[1:10000]; for i=ind, a(i)=log(i); end;...
e2=etime(clock,t1);
```

```
t1=clock; a=zeros(1,10000); ind=[1:10000];...
for i=ind, a(i)=log(i); end; e3=etime(clock,t1);
```

```
t1=clock; ind=[1:10000]; a=log(ind); e4=etime(clock,t1);
```

```
t1=clock; ind=[1:10000]; a=zeros(1,10000); a=log(ind); ...
e5=etime(clock,t1);
```

Los tiempos de computación para los diferentes métodos son:

```
86.17   86.56   2.42   0.27   0.28
```

Las causas de la disminución importante de tiempos es que en los primeros métodos, MATLAB tiene que recalcular la dimensión del vector cada pasada por el bucle (importancia de las inicializaciones), y además usa bucles `for`, que como se ha indicado, consumen mucho tiempo. Esto por supuesto no quiere decir que no deban usarse, pues habrá ocasiones en que no haya más remedio, pero siempre que haya una forma alternativa de hacerlo, ésta será preferible al uso de bucles.

- while

Permite implementar bucles condicionales. Su sintaxis es:

```
while expresion
    hacer algo;
end
```

La *expresión* es de la forma  $X$  operador  $Y$ , donde  $X$  e  $Y$  son escalares o expresiones que devuelven escalares y los operadores suelen ser operadores relacionales. En el siguiente ejemplo se busca una matriz aleatoria estable (parte real de autovalores negativa):

```
A = randn(2); % Genera numeros aleatorios con distribucion normal
while max(real(eig(A))) >= 0
    A=randn(2);
end;
eig(A)
```

Se puede usar el comando `break` para salir de un bucle en función de una determinada condición.

- if, else, elseif

La sintaxis es la siguiente:

```
if expresion 1
    hace algo
elseif expresion 2
    hace algo
else
    hace algo
end
```

`else` y `elseif` son opcionales.

### 1.13.3 Ficheros .m

MATLAB puede ejecutar programas que se encuentren almacenados en ficheros ASCII que pueden encontrarse en alguno de los subdirectorios indicados en el camino de búsqueda o bien en el subdirectorio de trabajo actual y tengan además extensión `.m`. Hay dos tipos de ficheros `.m`: *script files* y *function files*

#### Scripts

Son ficheros `.m` en los que se ponen secuencialmente comandos de MATLAB que se ejecutan en ese orden al introducir el nombre del fichero `.m` (sin extensión). Operan globalmente con

los datos que se encuentran en la memoria. Los ejemplos que ilustran estas notas son en sí *script-files*, pues llevan un conjunto de comandos MATLAB y comentarios.

### **funciones**

Son también ficheros `.m`, pero a diferencia de los anteriores, se le pueden pasar argumentos y pueden devolver resultados. Por tanto utilizan variables globales que se pasan por valor. La mayoría de los ficheros contenidos en los *toolboxes* son funciones. La sintaxis de todas las funciones almacenadas en ficheros `.m` es la siguiente:

```
function [out1,out2,...] = nombre_fichero (in1,in2,...)
% Comentarios adicionales para el help
comandos de MATLAB
return;
```

Una función puede tener múltiples parámetros de entrada y salida. Numerosos ejemplos de funciones serán utilizados en las demostraciones.

Para finalizar, comentar que existen una serie de utilidades a la hora de programar en MATLAB. Las más comunes son:

- **pause**: Para la ejecución hasta que se pulsa una tecla. Puede usarse para pausar la ejecución durante un número de segundos determinado, en lugar de esperar a que se pulse una tecla: `pause(n)`.
  
- **disp**: Muestra una cadena de caracteres por pantalla.
  
- **input**: Muestra una cadena de caracteres por pantalla y espera a que el usuario introduzca un valor, que generalmente será asignado a una variable.

## 1.14 RESUMEN DE LOS COMANDOS DE MATLAB

CARACTERES ESPECIALES	
=	Instrucción de asignación
[	Usado para formar vectores y matrices
]	Ver [
(	Precedencia aritmética
)	Ver (
.	Punto decimal
...	La instrucción continúa en la siguiente línea
,	Separa índices y argumentos de función
;	Acaba filas, suprime la impresión
%	Comentarios
:	Indexación, generación de vectores
!	Ejecuta instrucción del sistema operativo

VALORES ESPECIALES	
ans	Respuesta cuando no se asigna la expresión
eps	Precisión
pi	$\pi$
i,j	$\sqrt{-1}$
inf	$\infty$
NaN	No Número (Not-a -Number)
clock	Reloj
date	Fecha
flops	Número de operaciones
nargin	Número de argumentos de entrada de una función
narout	Número de argumentos de salida de una función

ARCHIVOS DE DISCO	
chdir	Cambiar de directorio
delete	Borrar archivo
diary	Diario de la sesión
dir	Directorio de archivos en el disco
load	Cargar variables de un archivo
save	Guardar variables en un archivo
type	Mostrar función o archivo
what	Mostrar archivos .m en el disco
fprintf	Escribir en un archivo
pack	Compactar memoria vía <b>save</b>

MATRICES ESPECIALES	
<code>compan</code>	Compañera
<code>diag</code>	Diagonal
<code>eye</code>	Identidad
<code>gallery</code>	Esotérica
<code>hadamard</code>	Hadamard
<code>hankel</code>	Hankel
<code>hilb</code>	Hilbert
<code>invhilb</code>	Inversa de Hilbert
<code>linspace</code>	Vectores igualmente espaciados
<code>logspace</code>	Vectores logarítmicamente espaciados
<code>magic</code>	Mágica cuadrada
<code>meshdom</code>	Dominio para puntos de malla
<code>ones</code>	Matriz constante de unos
<code>pascal</code>	Pascal
<code>rand</code>	Elementos aleatorios
<code>toeplitz</code>	Toeplitz
<code>vander</code>	Vandermonde
<code>zeros</code>	Matriz de ceros

MANIPULACIÓN DE MATRICES	
<code>rot90</code>	Rotación
<code>fliplr</code>	Invierte el orden de las columnas
<code>flipud</code>	Invierte el orden de las filas
<code>diag</code>	Diagonal
<code>tril</code>	Parte triangular inferior
<code>triu</code>	Parte triangular superior
<code>reshape</code>	Reordena una matriz en otra
<code>'</code>	Traspuesta
<code>:</code>	Convierte una matriz en una columna simple

FUNCIONES LÓGICAS Y RELACIONALES	
<code>any</code>	Condiciones lógicas
<code>all</code>	Condiciones lógicas
<code>find</code>	Encuentra índices de valores lógicos
<code>isnan</code>	Detecta NaNs
<code>finite</code>	Detecta infinitos
<code>isempty</code>	Detecta matrices vacías
<code>isstr</code>	Detecta variables de cadena
<code>strcmp</code>	Compara variables de cadena

CONTROL DE FLUJO	
<code>if</code>	Ejecuta instrucciones condicionalmente
<code>elseif</code>	Usado con <code>if</code>
<code>else</code>	Usado con <code>if</code>
<code>end</code>	Termina <code>if</code> , <code>for</code> , <code>while</code>
<code>for</code>	Repite instrucciones un número de veces
<code>while</code>	Repite instrucciones mientras una sentencia lógica sea verdadera
<code>break</code>	Sale de los bucles <code>for</code> y <code>while</code>
<code>return</code>	Salida desde funciones
<code>pause</code>	Pausa hasta que se pulse una tecla

TEXTO Y CADENAS	
<code>abs</code>	Convierte cadena en valores <b>ASCII</b>
<code>eval</code>	Evalúa texto como instrucciones
<code>num2str</code>	Convierte números en cadenas
<code>int2str</code>	Convierte enteros en cadenas
<code>setstr</code>	Indicador de cadenas
<code>sprintf</code>	Convierte números en cadenas
<code>isstr</code>	Detecta variables de cadena
<code>strcmp</code>	Compara variables de cadena
<code>hex2num</code>	Convierte cadenas hexadecimales en números

PROGRAMACIÓN Y ARCHIVOS <b>.m</b>	
<code>input</code>	Obtiene números desde el teclado
<code>keyboard</code>	Llamada al teclado como si fuera un archivo <b>.m</b>
<code>error</code>	Muestra mensaje de error
<code>function</code>	Define función
<code>eval</code>	Evalúa texto en variables
<code>feval</code>	Evalúa función dada por una cadena
<code>echo</code>	Permite mostrar las instrucciones en pantalla
<code>exist</code>	Comprueba si las variables existen
<code>casesen</code>	Sensibilidad a las mayúsculas
<code>global</code>	Define variables globales
<code>startup</code>	Archivo de inicialización
<code>getenv</code>	Accede a una variable de entorno
<code>menu</code>	Genera un menú
<code>etime</code>	Tiempo gastado

VENTANA ALFANUMÉRICA	
<code>clc</code>	Limpia pantalla
<code>home</code>	Mueve cursor al comienzo
<code>format</code>	Establece el formato de salida
<code>disp</code>	Muestra matriz o texto
<code>fprintf</code>	Imprime número formateado
<code>echo</code>	Permite la muestra de las instrucciones

GRÁFICOS	
<code>plot</code>	Gráfico lineal en el plano XY
<code>loglog</code>	Gráfico logarítmico en el plano XY
<code>semilogx</code>	Gráfico semilogarítmico
<code>semilogy</code>	Gráfico semilogarítmico
<code>polar</code>	Gráfico polar
<code>mesh</code>	Superficie de malla tridimensional
<code>contour</code>	Plano de contornos
<code>meshdom</code>	Dominio para gráficos de superficie
<code>bar</code>	Gráficos de barras
<code>stairs</code>	Gráficos de escaleras
<code>errorbar</code>	Añade barras de errores

ANOTACIÓN GRÁFICA	
<code>title</code>	Título
<code>xlabel</code>	Anotación en eje x
<code>ylabel</code>	Anotación en eje y
<code>grid</code>	Dibuja cuadriculado
<code>text</code>	Posiciona un texto arbitrariamente
<code>gtext</code>	Posiciona un texto con el ratón
<code>ginput</code>	input gráfico

CONTROL DE LA VENTANA GRÁFICA	
<code>axis</code>	Escalado manual de ejes
<code>hold</code>	Mantiene gráfico en pantalla
<code>shg</code>	Muestra la pantalla gráfica
<code>clf</code>	Limpia la pantalla gráfica
<code>subplot</code>	Divide la pantalla gráfica

FUNCIONES ELEMENTALES	
<code>abs</code>	Módulo complejo
<code>angle</code>	Argumento complejo
<code>sqrt</code>	Raíz cuadrada
<code>real</code>	Parte real
<code>imag</code>	Parte imaginaria
<code>conj</code>	Conjugado complejo
<code>round</code>	Redondeo al entero más cercano
<code>fix</code>	Redondeo hacia cero
<code>floor</code>	Redondeo hacia $-\infty$
<code>ceil</code>	Redondeo hacia $\infty$
<code>sign</code>	Función signo
<code>rem</code>	Resto
<code>exp</code>	Exponencial base $e$
<code>log</code>	Logaritmo natural
<code>log10</code>	Logaritmo base 10

FUNCIONES TRIGONOMÉTRICAS	
<code>sin</code>	Seno
<code>cos</code>	Coseno
<code>tan</code>	Tangente
<code>asin</code>	Arcoseno
<code>acos</code>	Arcocoseno
<code>atan</code>	Arcotangente
<code>atan2</code>	Arcotangente de x/y
<code>sinh</code>	Seno hiperbólico
<code>cosh</code>	Coseno hiperbólico
<code>tanh</code>	Tangente hiperbólica
<code>asinh</code>	Arcoseno hiperbólico
<code>acosh</code>	Arcocoseno hiperbólico
<code>atanh</code>	Arcotangente hiperbólica

FUNCIONES ESPECIALES	
<code>bessel</code>	Función de Bessel
<code>gamma</code>	Función gamma
<code>rat</code>	Aproximación racional
<code>erf</code>	Función de error
<code>inverf</code>	Inversa de la función de error
<code>ellipk</code>	Integral completa elíptica de primera especie
<code>ellipj</code>	Integral elíptica de Jacobi

DESCOMPOSICIONES Y FACTORIZACIONES	
<code>balance</code>	Forma equilibrada
<code>backsub</code>	Sustitución regresiva
<code>cdf2rdf</code>	Convierte diagonales complejas en diagonales reales
<code>chol</code>	Factorización de Cholesky
<code>eig</code>	Autovalores y autovectores
<code>hess</code>	Forma de Hessenberg
<code>inv</code>	Inversa
<code>lu</code>	Factores de la eliminación gaussiana
<code>nls</code>	Mínimos cuadrados con restricciones
<code>null</code>	Base ortonormal del núcleo
<code>orth</code>	Base ortonormal de la imagen
<code>pinv</code>	Pseudoinversa
<code>qr</code>	Factorización QR
<code>qz</code>	Algoritmo QZ
<code>rref</code>	Forma escalonada reducida por filas
<code>schur</code>	Descomposición de Schur
<code>svd</code>	Descomposición en valores singulares

CONDICIONAMIENTO DE MATRICES	
<code>cond</code>	Número de condición en la norma 2
<code>norm</code>	Norma 1, norma 2, norma de Frobenius, norma $\infty$
<code>rank</code>	Rango
<code>rcond</code>	Estimación de la condición (inverso)

FUNCIONES MATRICIALES ELEMENTALES	
<code>expm</code>	Matriz exponencial
<code>logm</code>	Matriz logaritmo
<code>sqrtn</code>	Matriz raíz cuadrada
<code>funm</code>	Función arbitraria de matriz
<code>poly</code>	Polinomio característico
<code>det</code>	Determinante
<code>trace</code>	Traza
<code>kron</code>	Producto tensorial de Kronecker

POLINOMIOS	
<code>poly</code>	Polinomio característico
<code>roots</code>	Raíces de polinomios - método de la matriz compañera
<code>roots1</code>	Raíces de polinomios - método de Laguerre
<code>polyval</code>	Evaluación de polinomios
<code>polyvalm</code>	Evaluación de polinomio matricial
<code>conv</code>	Multiplicación
<code>deconv</code>	División
<code>residue</code>	Desarrollo en fracciones parciales
<code>polyfit</code>	Ajuste por un polinomio

ANÁLISIS DE DATOS POR COLUMNAS	
<code>max</code>	Valor máximo
<code>min</code>	Valor mínimo
<code>mean</code>	Valor medio
<code>median</code>	Mediana
<code>std</code>	Desviación típica
<code>sort</code>	Ordenación
<code>sum</code>	Suma de elementos
<code>prod</code>	Producto de elementos
<code>cumsum</code>	Suma acumulativa de elementos
<code>cumprod</code>	Producto acumulativo de elementos
<code>diff</code>	Derivadas aproximadas
<code>hist</code>	Histogramas
<code>corrcoef</code>	Coefficientes de correlación
<code>cov</code>	Matriz de covarianza
<code>cplxpair</code>	Reordena en pares complejos

TRATAMIENTO DE SEÑALES	
<code>abs</code>	Módulo complejo
<code>angle</code>	Argumento complejo
<code>conv</code>	Convolución
<code>corrcoef</code>	Coefficientes de correlación
<code>cov</code>	Covarianza
<code>deconv</code>	Deconvolución
<code>fft</code>	Transformada rápida de Fourier
<code>fft2</code>	FFT 2-dimensional
<code>ifft</code>	FFT inversa
<code>ifft2</code>	FFT inversa 2-dimensional
<code>fftshift</code>	Cambia las dos mitades de un vector

INTEGRACIÓN NUMÉRICA	
<code>quad</code>	Función de integración numérica
<code>quad8</code>	Función de integración numérica

SOLUCIÓN DE ECUACIONES DIFERENCIALES	
<code>ode23</code>	Método Runge-Kutta de orden 2/3
<code>ode45</code>	Método Runge-Kutta-Fehlberg de orden 4/5

ECUACIONES NO LINEALES Y OPTIMIZACIÓN	
<code>fmin</code>	Mínimo de una función de una variable
<code>fmins</code>	Mínimo de una función de varias variables
<code>fsolve</code>	Solución de un sistema de ecuaciones no lineales (ceros de una función de varias variables)
<code>fzero</code>	Cero de una función de una variable

INTERPOLACIÓN	
<code>spline</code>	Spline cúbico
<code>table1</code>	Genera tablas 1-D
<code>table2</code>	Genera tablas 2-D

# Bibliografía

- [1] R.H. Bishop. *Modern Control Systems Analysis and Design Using MATLAB*. Addison-Wesley, 1993.
- [2] The MathWorks Inc. *CONTROL SYSTEM TOOLBOX User's Guide*. 1999.
- [3] The MathWorks Inc. *SIMULINK User's Guide, version 3*. 1999.
- [4] The MathWorks Inc. *Using MATLAB, version 5.3.1*. 1999.
- [5] K. Ogata. *Solving Control Engineering Problems with MATLAB, year=1994, publisher=Prentice Hall International Editions*.
- [6] B. Shahian and M. Hassul. *Control System Design using MATLAB, year=1993, publisher=Prentice Hall*.
- [7] K. Sigmon. *Introducción a MATLAB, Segunda Edición*. Department of Mathematics, U. Florida. Traducido del inglés por Celestino Montes, Dep. Matemática Aplicada II, U. Sevilla, 1992.