Lucía Agud Albesa
Mª Leonor Pla Ferrando

Matlab para matemáticas en ingenierías

EDITORIAL
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Los contenidos de esta publicación han sido revisados por el Departamento de Matemática Aplicada de la Universitat Politécnica de València

Colección Académica

Para referenciar esta publicación utilice la siguiente cita: AGUD ALBESA, L; PLA FERRANDO, Mª L (2015). *Matlab para matemáticas en ingenierías.* Valencia: Universitat Politècnica de València

distribución: Telf.: 963 877 012 / www.lalibreria.upv.es / Ref.: 0268_03_01_01

Imprime: Byprint Percom, sl

ISBN: 978-84-9048-421-0
Impreso bajo demanda

Impreso en España

Para todas las personas
que nos han ofrecido su apoyo día a día.

Contenido

Capítulo 1
Entorno de trabajo Matlab

1.1 Introducción

Este libro consta de dos partes, una dedicada al Álgebra: matrices, determinantes, sistemas de ecuaciones, etc; y otra enfocada al análisis matemático, donde se trabajarán funciones, expresiones algebraicas, ecuaciones, derivadas e integrales. El paquete matemático que se emplea es Matlab, cuyo nombre responde a las siglas de Matrix Laboratory.

Al abrir este paquete matemático se observa que la pantalla se subdivide en varias ventanas que se detallan a continuación:

- La ventana *Command Window* (ventana central) es donde se introducirán los comandos, variables e instrucciones a realizar. Es decir, la ventana donde se trabaja.
- La ventana *Current Folder* (izquierda superior) indica el contenido del directorio en el que se está trabajando, y que salvo cambio del mismo, es donde se irá guardando el archivo o las funciones que se hayan creado.
- La ventana de *Workspace* (derecha superior) es la ventana donde se indican las variables definidas en la sesión de trabajo o ya guardadas y cargadas de otras sesiones. Para borrar alguna de ellas, se usará el comando *>>clear nombredevariable* o se selecciona la variable en esa ventana y se suprime.

- La ventana de *Command History* muestra todos los comandos y órdenes que se han introducido, permitiendo recuperarlos o bien arrastrándolos a la *Command Window* (no ejecuta), o bien haciendo doble click sobre ellos (ejecuta).

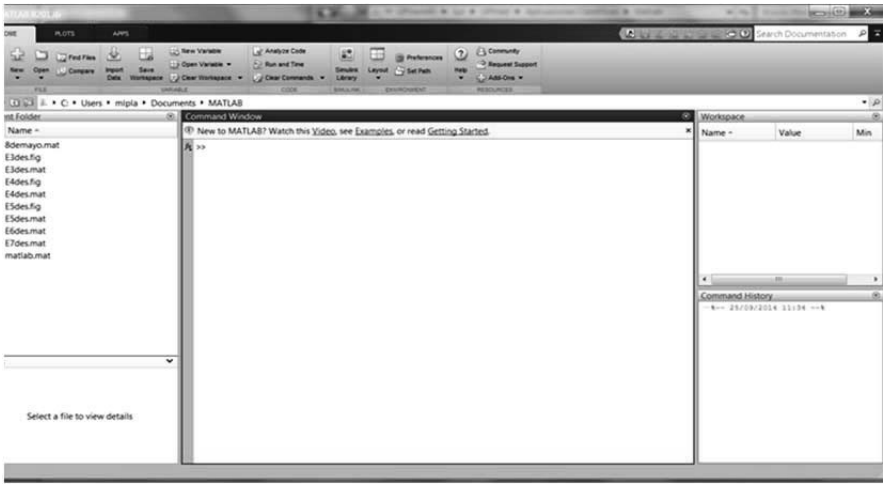

Figura 1.1 Pantalla de Matlab y ventanas

Al ejecutar el comando para representar una función, Matlab abre una nueva ventana de *Figure*, la cual se puede minimizar y mantener toda la sesión mientras se va actualizando, o bien cerrar.

Con el icono del folio o *New Script* de la barra de herramientas (primero de los iconos de la misma), también se abrirá una nueva ventana donde Matlab permite programar o crear funciones en ficheros con extensión .m.

La sesión de trabajo que se está realizando se puede guardar. Para ello, cuando se quiere comenzar a guardar se escribe (sin espacios en blanco en el nombre del fichero):

>>diary nombrefichero.txt

Y desde ahí hasta que se teclee *>>diary off*, guarda con el nombre indicado la sesión realizada. Si en cualquier momento se quiere volver a activar dicha sesión, bastará con poner de nuevo *>>diary on.* Por defecto, se guardará en la ruta que esté especificada en *Current Directory*.

En caso de querer guardar sólo las variables creadas en la sesión, bastará realizarlo con el comando *>>save*, o bien en *File >>Save Workspace*. Para recuperarlas en una nueva sesión de trabajo se cargan con el comando *>>load* y el nombre especificado cuando han sido guardadas.

1.2 Comandos o instrucciones en Matlab

Los comandos en Matlab se escriben en minúscula y sus argumentos entre paréntesis. Su escritura es en inglés y se ejecutan en cuanto se presiona la tecla *enter*.

Cuando hayan dudas de los argumentos de alguna instrucción, bastará con escribir en la ventana de comandos *>>help nombredelcomando*, como por ejemplo:

>>help gcd (y saldrá la ayuda de Matlab, en este caso para el cálculo del máximo común divisor).

Para recuperar alguno de los comandos introducidos, hay dos formas. Una, como ya se ha comentado, desde la ventana de *Command History;* la otra, con las teclas de desplazamiento. Las flechas de arriba y abajo, ↑, ↓ recuperan los comandos. Mientras que las de izquierda y derecha, ←, →, sirven para desplazarse dentro de la línea de edición para poder modificar las expresiones.

Para interrumpir el funcionamiento de una instrucción de MATLAB se pulsan las teclas *Control* y *C* a la vez y después *enter*. A veces, esta operación se deberá repetir.

La forma de salir de Matlab es o bien cerrar la ventana, o bien con los comandos *>>exit*, o *>>quit*. Y como siempre dándole al *enter* al final de cualquier instrucción para que así se ejecute.

Si no se quiere que la ejecución de un comando salga por pantalla, aunque sí que sea ejecutado, bastará con poner después del comando un *;*.

Al introducir una orden en la ventana *Command Window* que no quepa toda entera en una misma línea de instrucción, se puede terminar con ... y darle a *enter*. Matlab entiende que no ha terminado la instrucción y sin aparecer el *prompt* del sistema, >>, permite seguir escribiendo en la línea siguiente, ejecutando luego la instrucción al darle a *enter*:

```
>> A=[1 2 3 4 5 6 7 8 9;- 1 -2 -3 -4 -5 -6 -7 9 2;zeros(1,9);3*ones(1,9);...
9:-1:1]
A =
   1   2   3   4   5   6   7   8   9
  -1  -2  -3  -4  -5  -6  -7   9   2
   0   0   0   0   0   0   0   0   0
   3   3   3   3   3   3   3   3   3
   9   8   7   6   5   4   3   2   1
```

Si en una línea de instrucción se introduce el símbolo *%,* cambia el color del texto y automáticamente Matlab entiende que lo que se escribe, hasta pulsar *enter,* es un comentario y no debe ser ejecutado.

1.3 Variables y formatos

Cuando se ejecuta un comando sin dar nombre a la variable que se obtiene como resultado, Matlab lo asigna a una variable que él tiene en el sistema llamada *ans.* Nunca se podrá llamar a ninguna variable con este nombre, por tenerlo ya el programa asignado. En cada ejecución sin variable de salida, Matlab irá guardando en ella ese resultado, machacando el resultado anterior. Para guardar en una variable la operación realizada, se asignará con el =

>>a=gcd(3,12)

a=

3

Automáticamente en la ventana del *Workspace* aparecerá la variable *a* e indicará su tipo y dimensión. Para asignar a este nombre de variable otro valor, simplemente o se borra y vuelve a definir o se reasigna con otro valor. Matlab es sensible a mayúsculas y minúsculas; por lo tanto *a* y *A* son dos variables distintas.

No se puede nombrar a una variable cuyo nombre ya esté siendo utilizado por Matlab o bien en una función o comando, o bien para sus variables internas. Variables ya asignadas por Matlab son, entre otras:

i ó j	para la unidad imaginaria $i=\sqrt{-1}$
pi	para el valor de π
ans	para las variables de salida que no tengan asignación previa
eps	como valor es *2.2204e-016*.También puede utilizarse como comando (mirar *>>help eps* en caso de querer más información)

Matlab almacena internamente los números en formato de coma flotante normalizado; es decir, notación científica tal que, la parte entera es 0 y la primera cifra decimal es distinta de cero (0.0003 sería 0.3*e03). Generalmente, por pantalla muestra el resultado con el formato short, 4 dígitos decimales. Si interesa cambiar el formato, se consigue con el comando

>>format tipodeformato

Los formatos que existen en Matlab, y que usaremos generalmente, son:

FORMAT SHORT	*son 5 dígitos, contando parte entera y decimal.*
FORMAT LONG	*son 5 dígitos o 7*
FORMAT SHORTE	*son 5 dígitos en coma flotante normalizada*
FORMAT LONGE	*son 15 dígitos o 7 en coma flotante normalizada*
FORMAT SHORTG	*elige el mejor formato con 5 dígitos de salida (adecuado si trabajas en un vector con números de diferentes longitudes)*
FORMAT +	*saca por pantalla los signos, +, - y espacios en blanco*
FORMAT RAT	*pone el valor en forma racional.*

Si se quiere obtener más información sobre los formatos de Matlab, el programa la proporciona escribiendo

>> help format

1.4 Variables simbólicas y numéricas

Cuando en Matlab se quiere trabajar con una variable como tal, ha de definirse como simbólica. En caso contrario Matlab pediría valores para esa variable y daría un mensaje de error. Si se quiere introducir una función de forma que la variable *x* no tome valores concretos hasta cuando se requiera, existe el comando *>>sym*, o se declara la variable como simbólica de cualquiera de las dos formas siguientes:

>>x=sym('x'),

o bien *>>syms x*

De la segunda forma, es posible declarar todas las variables simbólicas en una misma sentencia, separadas por espacios en blanco.

Dar un valor concreto a esta variable para que deje de ser simbólica, y así conocer el valor numérico del resultado, se realiza con el comando *>>double.*

Ejemplo 1.1 Evaluar la función $f(x)=x^2-3x+2$ en $x=2, \pi/4$.

```
>>syms x
>>f=x^2-3*x+2;
>>subs(f,2)
ans=
        0
```

```
>>subs(f,pi/4)
ans =
 pi^2/16 - (3*pi)/4 + 2
>>double(ans)
ans =
   0.2607
```

> Nota: dependiendo de la versión de Matlab, esta última operación de sustitución puede hacerla de forma directa. Por ejemplo, en la versión R2011a; sin embargo en la versión R2013b hay que hacerlo como se ha indicado en el Ejemplo 1.1.

El comando *>>subs* siempre evalúa funciones de tipo simbólico y el resultado que devuelve es considerado también simbólico. Si se define la función de otra forma, por ejemplo con el comando *>>inline,* la instrucción *>>subs* no funciona; debe evaluarse con su nombre directamente o con el comando *>>feval*.

Ejemplo 1.2 Define la función $f(x)=x^2-3x+2$ con *inline* y evalúala en x=2,0.

```
>> clear x
>> g=inline('x^2-3*x+2')
g =
    Inline function:
    g(x) = x^2-3*x+2
>> g(2)
ans =
    0
>>feval(g,0)
ans=
   0
```

El comando *inline* será explicado con detalle más adelante.

Fracciones de decimales no exactos o valores irracionales

Es importante destacar qué ocurre cuando se trabaja con fracciones que no dan lugar a decimales exactos, por ejemplo fracciones del tipo 1/3, o radicales como $\sqrt{3}$, etc.

Matlab siempre guarda en memoria los datos en coma flotante, y por lo tanto redondea dichos números. Al efectuar operaciones con ellos, el resultado final puede verse afec-

tado de error. Una forma sencilla de solucionarlo es definir estos valores como simbólicos:

>>a=sym(1/3)

Ejemplo 1.3 Realizar la siguiente operación $1-\frac{\frac{1}{3}+\frac{1}{2}}{1-\frac{1}{6}}$ y comparar los resultados:

a) de forma numérica,
b) definiendo como simbólico los números.

Es evidente, sin más que realizar los cálculos, que esta operación debe dar 0:

>> 1-(1/3+1/2)/(1-1/6)

ans =

1.1102e-16

Sin embargo, Matlab no devuelve el valor 0 aunque sí un valor muy pequeño. Esto se debe al trabajo con decimales y su redondeo. La forma de solucionarlo sería:

>> a=sym(1/3);b=sym(1/6)
b =
1/6
>> 1-(a+1/2)/(1-b)
ans =

0

Otro problema de este estilo que se observa es, por ejemplo, si se quiere calcular algo tan sencillo como *sin(pi),* cuyo valor es 0. Matlab devuelve un valor muy pequeño, que se puede considerar como 0, pero no lo es. Por lo tanto, a la hora de resolver ecuaciones no detectará estos valores (una solución a esto se verá cuando se expliquen las funciones y el comando para resolver ecuaciones >>*solve*):

>>sin(pi)

ans =

1.2246e-16

En este caso, la opción para que evalúe bien vuelve a ser, definir este valor π como simbólico:

>>syms pi,sin(pi)

ans=

0

1.4.1 Salida matemática elegante por pantalla

Matlab escribe por pantalla las expresiones matemáticas tal cual deben ser introducidas:

```
>> syms x,
>> f = x^3-6*x^2+11*x-6
```

Sin embargo, mediante el comando *>>pretty(expression)* las muestra de la forma que uno escribe la expresión matemática:

```
>> pretty(f)
   3     2
  x  - 6 x  + 11 x – 6
```

Otro ejemplo sería:

```
>> syms x, y=1/(x+1)
 y =
1/(x + 1)
 >> pretty(y)
    1
  -----
  x + 1
```

1.4.2 Generar vectores o variables

Para generar una variable que tome valores en un rango elegido existen varias formas. Entre ellas cabe destacar las siguientes:

>>x=0:.2:12; ..... calcula un vector con $0 \leq x \leq 12$ donde los valores van de 0.2 en 0.2.

>>x=linspace(0,12,200); ..... calcula un vector con $0 \leq x \leq 12$ con 200 valores equiespaciados.

>>w=logspace(-1,3); ..... calcula 50 valores espaciados logarítmicamente entre 10^{-1} y 10^{3}.

> Nota: En los comandos *linspace* y *logspace*, el tercer argumento de entrada, correspondiente al número de puntos, es opcional. Los valores por defecto son 100 y 50, respectivamente. Asimismo, si no se especifica la distancia entre valores, el paso por defecto es 1.

Es importante destacar que las variables así definidas son vectores y, por ello, las operaciones algebraicas que las involucren deben realizarse con un punto delante: *./, .*, .^*. De esta forma, Matlab realiza los cálculos elemento a elemento, todo esto se explica con más detalle en el Capítulo 6.

1.4.3 Generar Matrices

Las matrices en Matlab se definen entre corchetes. Los elementos de cada fila pueden ir separados mediante comas o espacios en blanco. Las columnas van indicadas por punto y coma, *;*. Los elementos se pueden introducir de forma manual, con comandos o bien indicando el paso que lleva de un elemento al siguiente:

```
>> A=[1:2:9; 0 -2 4 6 -1;ones(1,5)]

A =
   1   3   5   7   9
   0  -2   4   6  -1
   1   1   1   1   1
```

Para saber más sobre matrices y matrices implementadas ya por Matlab, consultar la parte de Álgebra.

1.5 Funciones

Matlab posee muchas funciones implementadas. Para saber cuáles basta teclear *>>helpwin* y desde ahí acudir a *matlab\elfun,* donde mostrará las funciones que tiene definidas. Entre las más habituales destacan:

- Funciones seno y coseno: *sin(x), cos(x),sind(x),cosd(x)*
- Funciones seno y coseno hiperbólico: *sinh(x), cosh(x)*
- Función tangente: *tan(x)*
- Funciones arcoseno, arcocoseno y arcotangente: *asin(x), acos(x), atan(x)*
- Función exponencial, logaritmo neperiano y logaritmo decimal: *exp(x), log(x), log10(x)*
- Función valor absoluto: *abs(x)*
- Funciones hiperbólicas: *sinh(x),cosh(x),asinh(x)*, etc.

Nota: Matlab trabaja, salvo definición distinta de la variable, con valores numéricos. Concretamente, para las funciones trigonométricas, si no se declaran previamente como variables simbólicas, entiende que son valores numéricos y, por lo tanto, expresados en radianes. En caso de querer trabajar con valores expresados en grados, deben usarse las funciones:

>>sind, >>cosd

1.5.1 Crear funciones

Para crear funciones en Matlab existen varias formas. Aquí sólo se va a indicar cómo definirlas y evaluarlas. Para su representación gráfica consultar el tema de gráficos.

Se van a considerar dos tipos de funciones que engloban a todas, las simbólicas y las funciones a trozos. Existen varias posibilidades de definir funciones simbólicas.

A. Definición de función simbólica usando *Anonymus*

Esta definición se realiza mediante el símbolo @. Cuando las funciones son básicas de Matlab no hace falta indicar el argumento si este es *x*:

>>fun=@atan, ...Y para dibujarla *>> fplot(fun,[-2,2])*

>>fun2=@sin, ...Y para dibujarla *>> fplot(fun2,[-3,3])*

Sin embargo, si el argumento resulta ser composición de funciones o la función posee varios argumentos, entonces debe indicarse al principio, justo al lado de @, tal y como se muestra en el siguiente ejemplo

>> y=@(x) x^2+1;

Para evaluarla se puede hacer directamente asignándole un nombre:

```
>> y(1)
ans =
    2
>> y(0)
ans =
    1
```

O también con el comando *>>subs*

```
>> subs(y,0)
 ans =
 1
```

B. Definición de funciones simbólicas mediante el comando *>>inline*

Los argumentos se introducen entre comillas simples.

De esta forma se hallan los valores de la función sin más que calcular su imagen directamente o con el comando *>>feval*. Se detalla en el ejemplo siguiente

Ejemplo 1.4 Definir dos funciones cualesquiera con el comando >>*inline* y evaluarlas de forma distinta:

```
a)  >> f=inline('x^2-3*x+2')

f =
     Inline function:
    f(x) = x^2-3*x+2

>> f(2)

ans =

    0

b)  >> y=inline('x^2+1')

y =
     Inline function:

    y(x) = x^2+1

>> y(1)
ans =
     2
>> feval(y,1)
ans =

     2
```

Nota: Se recuerda que para evaluar funciones definidas mediante >>*inline* no se puede usar el comando >>*subs*.

C. Definición de funciones simbólicas definiendo la variable como simbólica

Para evaluar se utiliza el comando >>*subs*.

```
>> x=sym('x');p=2*x-5;

>> subs(p,1)

ans =

 -3
```

D.- Definición de función simbólica entre comillas simples

En este caso no hace falta declarar la variable. De nuevo es evaluada con el comando >>*subs*

```
>> y='x^2+1';
```

```
>> subs(y,0)
 ans =
 1
```

Para las funciones a trozos se indican aquí la forma vectorizada y mediante archivo .m; es decir, esta segunda desde el Editor de Matlab. Se ilustra mediante un ejemplo.

Ejemplo 1.5 Introducir, de varias formas la siguiente función a trozos:

$$f(x) = \begin{cases} x^2, & x < 1 \\ 1, & 1 \leq x < 3 \\ x+1, & x \geq 3 \end{cases}$$

1. Desde fichero .m

Esta definición debe hacerse desde la ventana del Editor de Matlab, icono de *NewScript* o más directamente en *New→Function.* Se crea con ello un archivo .m que, si no se indica nada, se guarda con el nombre que se le haya puesto a la función.

Es importante que si posteriormente se modifica, se tenga cuidado de no cambiarle el nombre, ya que la llamada para ser ejecutada se hace por el nombre del fichero.

Para evaluarla basta referirse a ella con su nombre.

Si se realiza para la función del Ejemplo 1.5, quedaría

```
function y=funcion2(x)
if x<1
   y=x^2;
elseif 1<=x&&x<3
   y=1;
else
   y=x+1;
end
```

y para evaluarla

```
>>funcion2(2)   % para evaluarla
ans=
1
```

2. Vectorizada

- Para introducir y evaluar una función así definida es conveniente usar el comando *>>inline*:

```
>>f=inline('(x.^2).*(x<1)+1.*((1<=x)&(x<=3))+(x+1).*(3<x)')
f =
     Inline function:
     f(x) = (x.^2).*(x<1)+1.*((1<=x)&(x<=3))+(x+1).*(3<x)
>> f(3)
ans =
     1
```

- También puede definirse mediante *Anonymus,* en este caso hay que evaluarla, de nuevo, llamándola por el nombre que tiene asignado:

```
>> f=@(x)(x.^2).*(x<1)+1.*((1<=x)&(x<=3))+(x+1).*(3<x)
f =
    @(x)(x.^2).*(x<1)+1.*((1<=x)&(x<=3))+(x+1).*(3<x)
>> f(-3)
ans =
     9
>> f(4)
ans =
     5
>> f(2)
ans =
     1
```

Para representarlas gráficamente, dirigirse al Capítulo 2, donde se indica cómo dibujar cada una de las posibles definiciones.

Para seguir leyendo haga click aquí