MATLAB

En Matlab existen dos formas de trabajar:

a) En el "espacio de trabajo" o pantalla inicial de Matlab, como calculadora o mediante bucles. De esta manera se ejecutan las órdenes de forma inmediata y los resultados se visualizan por pantalla, si no se pone ";" al final de cada una de ellas.

b) En editor, por medio de programas o ficheros de función. Para crear un programa, o un fichero, se elige la opción NEW del menú FILE, y se toma M-FILE. De esta forma se crea un fichero con extensión M, que Matlab reconoce como ejecutable. Para guardar un fichero se utiliza la expresión SAVE AS o SAVE del menú FILE del editor (MEDIT).

Tanto en el "espacio de trabajo" como en el editor, el comando "%" indica que a partir de dicha orden, lo que aparece en la misma línea es un comentario.

Las operaciones básicas que ofrece Matlab son: suma, resta, producto, división y potenciación.

```
3+4-6*2

ans =

  -5
```

También se puede asignar una variable:

```
y=3+4-6*2

y =

  -5
```

En caso de que no se asigne, Matlab toma la variable ANS por defecto:

```
3^2-7-12/2*3

ans =

  -16
```

Las expresiones se evalúan de izquierda a derecha, siguiendo el orden conocido de preferencia de las operaciones matemáticas.

Para recordar el nombre de las variables utilizadas en el "espacio de trabajo" se utiliza la orden WHO. Si además del nombre, deseamos saber otras características de las variables, se utiliza la orden WHOS:

```
who

Your variables are:

ans     y

whos

  Name      Size          Bytes  Class

  ans       1x1               8  double array
  y         1x1               8  double array
```

Grand total is 2 elements using 16 bytes

La orden CLEAR sirve para borrar las variables del "espacio de trabajo". Así

clear y

elimina la variable "y", con lo cual al volver a preguntar por las variables

whos

Name	Size	Bytes	Class
ans	1x1	8	double array

Grand total is 1 elements using 8 bytes

observamos que sólo queda la variable ANS.

Cuando una orden es demasiado larga, se escriben tres puntos "..." seguido de la orden ENTER para continuar en la línea siguiente.

Para ayudar a encontrar órdenes, Matlab proporciona el comando HELP, que se puede utilizar de dos formas, dependiendo de que sepamos o no, el nombre del tema sobre el que necesitamos ayuda.

help

HELP topics:

matlab\general	-	General purpose commands.
matlab\ops	-	Operators and special characters.
matlab\lang	-	Programming language constructs.
matlab\elmat	-	Elementary matrices and matrix manipulation.
matlab\elfun	-	Elementary math functions.
matlab\specfun	-	Specialized math functions.
matlab\matfun	-	Matrix functions - numerical linear algebra.
matlab\datafun	-	Data analysis and Fourier transforms.
matlab\polyfun	-	Interpolation and polynomials.
matlab\funfun	-	Function functions and ODE solvers.
matlab\sparfun	-	Sparse matrices.
matlab\graph2d	-	Two dimensional graphs.
matlab\graph3d	-	Three dimensional graphs.
matlab\specgraph	-	Specialized graphs.
matlab\graphics	-	Handle Graphics.
matlab\uitools	-	Graphical user interface tools.
matlab\strfun	-	Character strings.
matlab\iofun	-	File input/output.
matlab\timefun	-	Time and dates.
matlab\datatypes	-	Data types and structures.
matlab\dde	-	Dynamic data exchange (DDE).
matlab\demos	-	Examples and demonstrations.
toolbox\symbolic	-	Symbolic Math Toolbox.
toolbox\signal	-	Signal Processing Toolbox.
toolbox\control	-	Control System Toolbox.
control\obsolete	-	(No table of contents file)
toolbox\local	-	Preferences.

For more help on directory/topic, type "help topic".

Si tecleeamos HELP, Matlab facilita un listado general de los diferentes temas generales. Al introducir HELP\NOMBRE obtenemos la ayuda correspondiente al tema especificado. Así, si queremos obtener información sobre operadores:

help matlab\ops

Operators and special characters.

Arithmetic operators.

plus	-	Plus	+
uplus	-	Unary plus	+
minus	-	Minus	-
uminus	-	Unary minus	-
mtimes	-	Matrix multiply	*
times	-	Array multiply	.*
mpower	-	Matrix power	^
power	-	Array power	.^
mldivide	-	Backslash or left matrix divide	\
mrdivide	-	Slash or right matrix divide	/
ldivide	-	Left array divide	.\
rdivide	-	Right array divide	./
kron	-	Kronecker tensor product	kron

Relational operators.

eq	-	Equal	==
ne	-	Not equal	~=
lt	-	Less than	<
gt	-	Greater than	>
le	-	Less than or equal	<=
ge	-	Greater than or equal	>=

Logical operators.

and	-	Logical AND	&
or	-	Logical OR	|
not	-	Logical NOT	~
xor	-	Logical EXCLUSIVE OR	
any	-	True if any element of vector is nonzero	
all	-	True if all elements of vector are nonzero	

Special characters.

colon	- Colon	:
paren	- Parentheses and subscripting	()
paren	- Parentheses and subscripting	()
paren	- Brackets	[]
paren	- Braces and subscripting	{}
paren	- Braces and subscripting	{}
punct	- Decimal point	.
punct	- Structure field access	.
punct	- Parent directory	..
punct	- Continuation	...
punct	- Separator	,
punct	- Semicolon	;
punct	- Comment	%
punct	- Invoke operating system command	!
punct	- Assignment	=
punct	- Quote	'

transpose	- Transpose	.'
ctranspose	- Complex conjugate transpose	'
horzcat	- Horizontal concatenation	[,]
vertcat	- Vertical concatenation	[;]
subsasgn	- Subscripted assignment	( ),{ },.
subsref	- Subscripted reference	( ),{ },.
subsindex	- Subscript index	

Bitwise operators.

bitand	-	Bit-wise AND.
bitcmp	-	Complement bits.
bitor	-	Bit-wise OR.
bitmax	-	Maximum floating point integer.
bitxor	-	Bit-wise XOR.
bitset	-	Set bit.
bitget	-	Get bit.
bitshift	-	Bit-wise shift.

Set operators.

union	-	Set union.
unique	-	Set unique.
intersect	-	Set intersection.
setdiff	-	Set difference.
setxor	-	Set exclusive-or.
ismember	-	True for set member.

See also ARITH, RELOP, SLASH.

Otra forma de obtener ayuda en línea, es mediante la orden, LOOKFOR, acompañada de una palabra clave, que no necesita ser una orden de Matlab:

lookfor real

REALMAX Largest positive floating point number.
REALMIN Smallest positive floating point number.
ISREAL True for real array.
REAL Complex real part.
CDF2RDF Complex diagonal form to real block diagonal form.
RSF2CSF Real block diagonal form to complex diagonal form.
A2ODE Stiff problem, linear with real eigenvalues (A2 of EHL).
A3ODE Stiff problem, linear with real eigenvalues (A3 of EHL).
D1ODE Stiff problem, nonlinear with real eigenvalues (D1 of EHL).
RCEPS Real cepstrum.
BALREAL Gramian-based balancing of state-space realizations.
CANON Canonical state-space realizations.
COMPBAL Balancing for SIMO state-space realizations in
COMPREAL Companion realization of SIMO transfer functions
MINREAL Minimal realization and pole-zero cancellation.
DBALREAL Discrete balanced state-space realization and model reduction.

Además de la ayuda en línea, existe la que se obtiene al elegir la opción HELP de la barra de menús.

NÚMEROS: FORMATOS DE VISUALIZACIÓN.

Matlab reconoce cualquier clase de número, sea real o imaginario:

```
sqrt(-16)

ans =

    0+ 4.0000i
```

Por defecto utiliza el formato corto -FORMAT SHORT, que consta de cuatro dígitos decimales, pero también dispone de otros tipos de formatos:

```
format long

sqrt(-16)

ans =

        0 + 4.00000000000000i
```

que consta de 16 dígitos. Otros tipos de formato son:

FORMAT RAT que proporciona la aproximación racional

FORMAT SHORT E que consta de 5 dígitos más exponente

FORMAT LONG E con 16 dígitos más exponente

FORMAT BANK que proporciona 2 dígitos decimales.

VARIABLES.

Para escribir una variable en Matlab, se puede utilizar una sola palabra o varias unidas mediante un guión "_" y no debe contener signos de puntuación ni sobrepasar 31 caracteres. Distingue entre mayúsculas y minúsculas, así la variable "AB_12" es distinta de "ab_12".

En Matlab no se necesita asignar un tipo predeterminado a las variables, una vez definidas, las toma siempre de la misma forma.

Existen una serie de variables especiales, tales como:

PI	razón entre la circunferencia y su diámetro
EPS	mínimo número que sumado a 1, da como resultado el siguiente mayor que 1
NAN	magnitud no numérica
REALMIN	el número real positivo más pequeño que es utilizable
REALMAX	el número real positivo más grande que es utilizable.

FUNCIONES MATEMÁTICAS.

La mayoría de las funciones que Matlab proporciona se utilizan de la misma forma a que estamos acostumbrados. Indicamos a continuación algunas de dichas funciones:

ABS(x)	Proporciona el valor absoluto o medida de un número complejo
ACOS(X)	Inversa del coseno
ACOSH(x)	Inversa del coseno hiperbólico
ANGLE(x)	Ángulo de un número complejo
ASIN(X)	Inversa del seno
ATAN2(x,y)	Inversa de la tangente en los cuatro cuadrantes
CEIL(x)	Redondea hacia más infinito
COS(x)	Coseno
EXP(x)	Exponencial de base "e"
FIX(x)	Redondea hacia cero
FLOOR(x)	Redondea hacia menos infinito
IMAG(x)	Parte imaginaria de un número complejo
LOG(x)	Logaritmo de base "e"
LOG10(x)	Logaritmo decimal
REAL(x)	Parte real de un número complejo
REM(x,y)	Resto de la división de x/y
ROUND(x)	Redondea hacia el entero más próximo
SIGN(x)	Devuelve el signo del argumento: 1, si x>0; -1, si x<0; 0, si x=0
SINH(x)	Seno hiperbólico
SQRT(x)	Raíz cuadrada
TANH(x)	Tangente hiperbólica.

Aquí hemos indicado una parte de dichas funciones importantes, aunque existen más.

Como hemos comentado en el apartado de números, Matlab reconoce números complejos y las funciones REAL, IMAG, ABS y ANGLE son útiles para la conversión entre las formas polar y rectangular.

VECTORES.

Hasta ahora hemos trabajado con escalares, pero también se puede trabajar en forma vectorial. Existen varias maneras de introducir vectores:

a) Indicando, entre corchetes, las componentes del vector

```
v1=[1 3 5 6]

v1 =

   1   3   5   6

v2=[2;3;5;7]

v2 =

   2
   3
   5
   7
```

b) Escribiendo la componente inicial, la final y la distancia, entre ":"

```
v3=-1:2:13

v3 =

  -1   1   3   5   7   9  11  13
```

c) Por medio de la orden LINSPACE(primera, última, número de componentes)

```
v4=linspace(-1,pi,4)

v4 =

  -1.0000   0.3805   1.7611   3.1416
```

d) La orden LOGSPACE crea un vector cuyas componentes están distanciadas en forma logarítmica

```
v5=logspace(0,2,6)

v5 =

   1.0000   2.5119   6.3096  15.8489  39.8107 100.0000
```

que como observamos comienza en 10^0 y termina en 10^2, teniendo seis componentes.

Si queremos expresar la componente de un vector se utiliza el nombre del vector y la componente o componentes entre paréntesis.

```
v1(3)

ans =

   5

v5(1:3)

ans =

   1.0000   2.5119   6.3096
```

```
v5(2:2:6)

ans =

  2.5119  15.8489  100.0000
```

Cuando dos vectores tienen la misma dimensión se pueden sumar, restar, multiplicar dividir y efectuar operaciones con potencias:

```
v1+v4

ans =

       0  3.3805  6.7611  9.1416

3*v1-2

ans =

   1   7  13  16

v1.^2

ans =

   1   9  25  36
```

Observamos que se puede elevar un vector a una potencia, pero se hace a través de la operación especial ".^", que calcula la potencia de cada una de las componentes del vector.

También se pueden dividir, componente a componente, dos vectores por medio del operador "./" (división a derecha), o por medio de ".\" (división à izquierda). Lo mismo se puede decir de la multiplicación entre vectores de la misma dimensión, para la que existe la operación especial ".*". Ejemplos:

```
v1./v4

ans =

-1.0000e+000  7.8837e+000  2.8392e+000  1.9099e+000

v4.\v1

ans =

-1.0000e+000  7.8837e+000  2.8392e+000  1.9099e+000

v4.*v1

ans =

-1.0000e+000  1.1416e+000  8.8053e+000  1.8850e+001
```

La orden " ' ", sirve para trasponer vectores. Si el vector está formado por números complejos el resultado es el traspuesto conjugado. Si solamente deseamos obtener el vector traspuesto deberemos aplicar la orden " .' ". En el caso de componentes reales ambas órdenes proporcionan el mismo resultado:

```
v1'

ans =

    1
    3
    5
    6

v1.'

ans =

    1
    3
    5
    6

v6=[1-i 2+3i -i];

v6'

ans =

  1.0000+ 1.0000i
  2.0000- 3.0000i
     0+ 1.0000i

v6.'

ans =

  1.0000- 1.0000i
  2.0000+ 3.0000i
     0- 1.0000i
```

Pero los vectores también pueden estar formados por letras. Es lo que se conoce como 'cadenasde caracteres' o simplemente 'cadenas'. El texto de una cadena tiene que ir entre comillas " ' ' ", y se maneja igual que un vector fila:

```
t='Esto es un tutorial de Matlab'

t =

Esto es un tutorial de Matlab
```

Las cadenas se direccionan igual que los vectores:

```
u=t(12:19)

u =

tutorial
```

nos devuelve las letras situadas entre los lugares 12 y 19.

Al igual que las matrices están formadas por varias filas de vectores, las cadenas de caracteres pueden tener múltiples filas, pero cada fila debe tener el mismo número de columnas.

```
t1=['Esto es una comprobación'
    'de que se pueden formar '
    'varias filas            ']

t1 =

Esto es una comprobación
de que se pueden formar
varias filas

s='abcde'

s =

abcde

ab=abs(s)

ab =

   97   98   99  100  101

m=s+5

m =

  102  103  104  105  106
```

Como las cadenas de caracteres son vectores con comillas, se les puede aplicar lo indicado para vectores, pero una vez que se aplica alguna operación matemática sobre una cadena, ésta ya no se visualiza como cadena sino como un vector de números en el código ASCII.

OPERADORES

Además de las operaciones matemáticas, Matlab permite operaciones relacionales y lógicas. El resultado de todas las expresiones relacionales y lógicas produce 1 si es verdadera y 0 si es falsa.

Los operadores *relacionales* son los siguientes:

<	menor que
<=	menor o igual que
>	mayor que
>=	mayor o igual que
==	igual a
~=	no igual a

Vamos a considerar algunos ejemplos de aplicación:

```
abs(v1)>=3

ans =

   0   1   1   1
```

indica, por medio del número 1, las componentes cuyo valor absoluto es mayor o igual que 3.

```
vector=v1(abs(v1)>=3)

vector =

   3   5   6
```

devuelve el valor de las componentes del vector v1 que verifican la condición de ser mayores o iguales que 3.

```
v2=[6 4 3 1];

logica=v1==v2

logica =

   0   0   0   0
```

confirma que ninguna de las componentes de los vectores v1 y v2 coinciden.

```
logica2=v1<=v2

logica2 =

   1   1   0   0
```

nos indica que las dos primeras componentes del vector v1 son menores que sus correspondientes del vector v2.

```
v3=v2(v1>=v2)

v3 =

   3   1
```

proporciona el valor de las componentes del vector v2 correspondientes a las componentes del v1 que cumplen la condición de ser mayores o iguales que las de v2.

Los operadores lógicos que más se utilizan son:

&	y
|	o
~	no

se pueden unir operadores relacionales y lógicos. Ejemplo:

```
operador=(v1>1)&(v1<5)

operador =

   0   1   0   0
```

En Matlab, los operadores aritméticos son los que tienen mayor orden de preferencia (comenzando por exponenciación y trasposición, multiplicación y división, suma y resta), seguidos de los operadores relacionales y por último, los operadores lógicos.

MATRICES

Matlab trabaja con matrices de la misma forma que lo hace con vectores. Para introducir una matriz se ponen entre corchetes los números por filas:

```
M=[1 3 4;2 3 5;3 4 6]

M =

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

Otra forma de introducir una matriz, es, mediante vectores o combinación de vectores y números:

```
M1=[v1;v2;3 4 8 5]

M1 =

   1   3   5   6
   6   4   3   1
   3   4   8   5
```

Si se quiere conocer un elemento de una matriz, se pone el nombre de la matriz acompañado de la fila y columna que ocupa el elemento entre paréntesis

```
M1(2,4)

ans =

   1
```

Si queremos visualizar una fila o una columna, se ponen ":" en el lugar de columna o fila, y se escribe en su lugar correspondiente el número de fila o de columna que deseemos

```
M(:,2)

ans =

   3
   3
   4
```

proporciona la segunda columna de la matriz M

```
M1(1,:)

ans =

   1   3   5   6
```

escribe la primera fila de la matriz M1.

Además, se pueden entresacar filas y columnas consecutivas o no, indicando en el lugar de las filas, las que deseemos y de forma idéntica en el lugar de las columnas:

```
M1([1 3],2:3)
```

```
ans =

   3   5
   4   8
```

proporciona la matriz intersección de las filas 1 y 3 con las columnas 2 y 3.

Si queremos obtener la matriz identidad de orden "n" es suficiente escribir la orden EYE acompañada del número de orden:

```
eye(4)

ans =

   1   0   0   0
   0   1   0   0
   0   0   1   0
   0   0   0   1
```

La matriz de ceros de cualquier orden se obtiene mediante la orden ZEROS seguida del número de orden. Así:

```
zeros(3)

ans =

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

y la orden ONES(n) proporciona una matriz compuesta por unos

```
ones(4)

ans =

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

Las operaciones que se pueden efectuar con matrices son: suma, resta, multiplicación, división, potenciación (estas tres últimas operaciones se pueden realizar en forma matricial o elemento a elemento), y trasposición.

```
M2=[1 3 5;3 1 0;4 4 -1]

M2 =

   1   3   5
   3   1   0
   4   4  -1

M+M2

ans =

   2   6   9
   5   4   5
   7   8   5
```

2*M2-M

ans =

```
   1   3   6
   4  -1  -5
   5   4  -8
```

M*M2

ans =

```
  26  22   1
  31  29   5
  39  37   9
```

donde "*"es la multiplicación matricial

M.*M2

ans =

```
   1   9  20
   6   3   0
  12  16  -6
```

que es la multiplicación elemento a elemento

M*M1

ans =

```
  31  31  46  29
  35  38  59  40
  45  49  75  52
```

se puede efectuar la multiplicación cuando el número de columnas de la primera matriz es igual que el número de filas de la segunda, en caso contrario da mensaje de error

M1*m

_??? Error using ==> *

Inner matrix dimensions must agree.

Se puede obtener la diagonal, en forma de vector, de una matriz cuadrada mediante la orden DIAG(nombre)

diag(M)

ans =

```
   1
   3
   6
```

Si aplicamos dos veces esta orden, obtenemos la diagonal en una matriz, cuyos restantes elementos son iguales a cero:

```
ans =

   1   0   0
   0   3   0
   0   0   6
```

Para calcular el número de filas y de columnas de una matriz se utiliza la orden SIZE(nombre). Si tecleamos SIZE(nombre,1) o SIZE(nombre,2), obtenemos el número de filas o de columnas, respectivamente.

```
size(M1)

ans =

   3   4

size(M1,1)

ans =

   3

size(M1,2)

ans =

   4
```

La orden LENGTH(nombre) visualiza la mayor dimensión de filas o de columnas de la matriz:

```
length(M1)

ans =

   4
```

POLINOMIOS

En Matlab se puede trabajar con polinomios, que se representan como un vector fila con los coeficientes en orden descendiente. Así, el polinomio x^4+2x^2+1 se escribe

```
p=[1 0 2 0 1]

p =

   1   0   2   0   1
```

Para encontrar los ceros de un polinomio, existe la orden ROOTS

```
r=roots(p)

r =

  0.0000+ 1.0000i
  0.0000- 1.0000i
  0.0000+ 1.0000i
  0.0000- 1.0000i
```

Conocidas las raíces de un polinomio, éste se puede reconstruir utilizando el comando POLY(raices)

```
pp=poly(r)

pp =

   1.0000   0.0000   2.0000   0.0000   1.0000
```

Se pueden multiplicar (dividir) dos polinomios por medio de los comandos CONV(p,q) (DECONV(p,q))

```
q=[3 -2 0 0 0 12];

conv(p,q)

ans =

   3   -2   6   -4   3   10   0   24   0   12

[c,r]=deconv(q,p)

c =

   3   -2


r =

   0   0   -6   4   -3   14
```

obtenemos el cociente y el resto de la división.

Para sumar polinomios tienen que tener la misma dimensión, en caso contrario se tienen que añadir ceros hasta completar la dimensión.

```
d=p+q
_??? Error using ==> +
Matrix dimensions must agree.

d=[0 p]+q

d =

   3   -1   0   2   0   13
```

También se puede, mediante la orden POLYDER, calcular la derivada de un polinomio

```
polyder(q)

ans =

   15   -8   0   0   0
```

MATEMATICA SIMBOLICA

En las últimas versiones de Matlab existe la posibilidad de efectuar cálculos de una forma simbólica. Para ello se utilizan objetos simbólicos que representan números, funciones, operadores y variables. La forma de crear un objeto simbólico es mediante comillas " ' ' ", o por la expresión SYM(nombre).

```
m=[1 3;5 7]

m =

   1   3
   5   7

M=sym(m)

M =

[ 1, 3]
[ 5, 7]
```

crea la matriz simbólica de m.

Se pueden derivar e integrar funciones por medio de cálculo simbólico. Así

```
f='cos(x)-log(x)'

f =

cos(x)-log(x)

diff(f)

ans =

-sin(x)-1/x
```

proporciona la primera derivada de la función "f"

```
diff(f,3)

ans =

sin(x)-2/x^3
```

proporciona la tercera derivada de dicha función.

```
int(f)

ans =

sin(x)-x*log(x)+x
```

calcula la integral, si derivamos dicha integral obtendremos la función inicial

```
diff('sin(x)-x*log(x)+x')

ans =
```

```
cos(x)-log(x)
```

Para representar una función simbólica se utilizan las órdenes EZPLOT(nombre) o FPLOT((nombre),intervalo)

```
fplot(f,[1 3])
```

o

```
ezplot(diff(f))
```

INTRODUCCION Y SALIDA DE DATOS

Para introducir datos por pantalla se utiliza la orden INPUT

```
y=input(' Dame la variable inicial ')

 Dame la variable inicial 3

y =

   3
```

Otra posibilidad de introducir datos, es por medio del comando SPRINTF que permite incluir datos numéricos en el texto inicial:

```
i=2

i =

   2

x=sprintf(' Dame la %dª variable inicial ', i)

x =

 Dame la 2ª variable inicial 2

x =

   2
```

x = sprintf('Dame la %d ª var. inicial', i)
a = input(x)

Para sacar datos por pantalla se utiliza la orden DISP o FPRINTF, ésta permite dar formato a los datos de salida

```
disp(y)
   3

disp(' y es la variable obtenida')

 y es la variable obtenida


fprintf( 'la variable obtenida es %3.3f',i)
la variable obtenida es 2.000
```