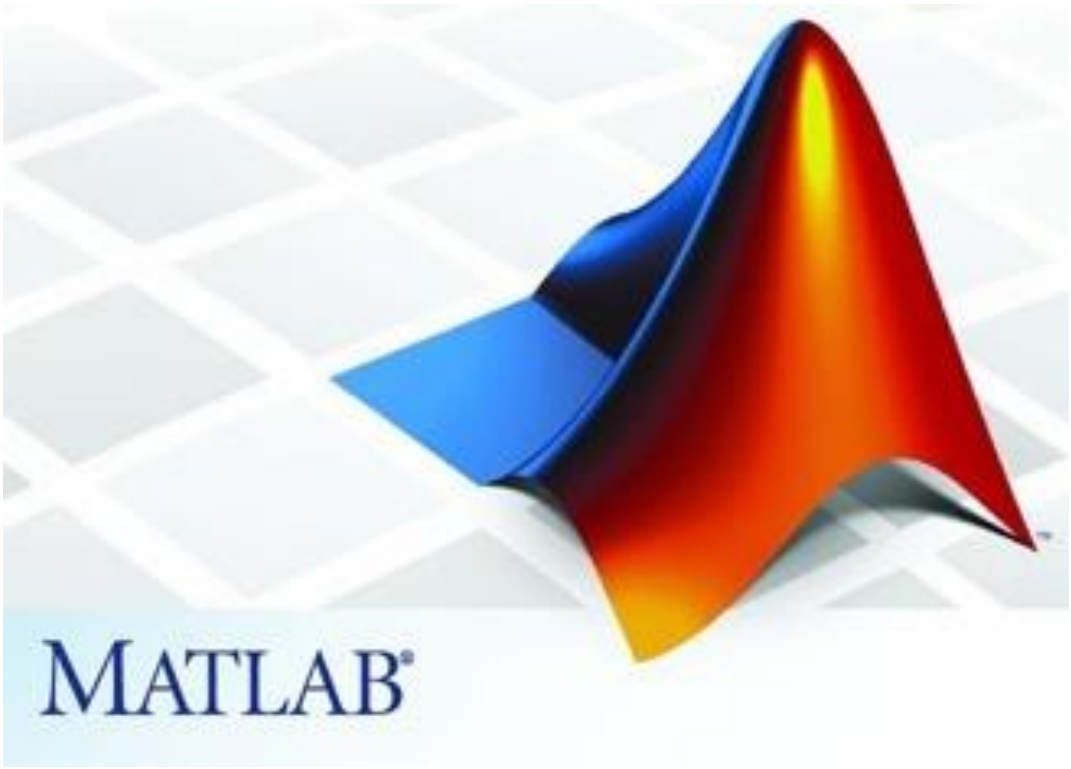
MANUAL BÁSICO



OPERACIONES BÁSICAS

De acuerdo a su orden de precedencia:

Operación	Símbolo	Expresión en MATLAB
Potencia	^	a ^ b
Multiplicación	*	a * b
División	/	a / b
Suma	+	a + b
Resta	-	a - b

Matlab no tiene en cuenta los espacios. Para que Matlab evalúe la línea pero que no escriba la respuesta, basta escribir punto y coma (;) al final de la sentencia. Si la sentencia es demasiado larga para que quepa en una sola línea podemos poner tres puntos (…) seguido de la tecla “Intro” para indicar que continúa en la línea siguiente.

FORMATOS DE VISUALIZACIÓN DE NÚMEROS

Matlab no cambia la representación interna de un numero cuando se escogen distintos formatos, solo modifica la forma de visualizarlo.

Tipo	Resultado
format short	Formato coma fija, 4 dígitos después de la coma (por defecto).
format long	Formato coma fija, 14 o 15 dígitos después de la coma.
format short e	Formato coma flotante, 4 dígitos después de la coma.
format long e	Formato coma flotante, 14 o 15 dígitos después de la coma.
format short g	Entre coma fija o flotante 4 dígitos después de la coma.
format long g	Entre coma fija o flotante, 14 o 15 dígitos después de la coma.
format short eng	Notación científica con 4 dígitos significantes y un exponente de 3.
format long eng	Notación científica con 16 dígitos significantes y un exponente de 3.
format bank	Formato coma fija con 2 dígitos después de la coma.
format hex	Hexadecimal .
format rat	Aproximación racional.
format +	Positivo, negativo o espacio en blanco.

VARIABLES

Las variables son sensibles a las mayúsculas, deben comenzar siempre con una letra, no pueden contener espacios en blanco y pueden nombrarse hasta con 63

caracteres (en versiones anteriores no permitía tantos caracteres). Si se nombra una variable con más de 63 caracteres truncará el nombre de dicha variable.

Variable	Definición	Valor
ans	Variable usada por defecto para almacenar el ultimo resultado	???
pi	Razón de una circunferencia a su diámetro	3.1416
eps	Número más pequeño, tal que cuando se le suma 1, crea un numero en coma flotante en el computador mayor que 1	2.2204e-016
inf	Infinito	Inf
nan	Magnitud no numérica	NaN
i y j	i=j=$\sqrt{-1}$	0+1.0000i
realmin	El número real positivo más pequeño que es utilizable	2.2251e-308
realmax	El número real positivo más grande que es utilizable	1.7977e+308
clear	Borra todas las variables del espacio de trabajo	
clc	Borra lo que hay en la ventana de comandos	
clock	Muestra el año, mes, día, hora, minutos y segundos	
date	Muestra la fecha, día, mes, año	
calendar	Muestra el calendario del mes actual	

Para borrar todas las variables del espacio de trabajo, se usa “clear”.

Para borrar lo que hay en la ventana de comandos, se usa “clc”.

APROXIMACIONES

Función	Definición
ceil (x)	Redondea hacia el infinito
fix (x)	Redondea hacia cero
floor (x)	Redondea hacia menos infinito
round (x)	Redondea hacia el entero más próximo

TRIGONOMETRÍA

Función	Definición
… (x)	Función trigonométrica con el ángulo expresado en radianes
sin (x)	Seno (radianes)
cos (x)	Coseno
tan (x)	Tangente
csc (x)	Cosecante
sec (x)	Secante
cot (x)	Cotangente

… d (x)	Función trigonométrica con el ángulo expresado en grados
sind (x) cosd (x) tand (x) cscd (x) secd (x) cotd (x)	Seno (grados) Coseno Tangente Cosecante Secante Cotangente
… h (x)	Función trigonométrica hiperbólica con el ángulo expresado e radianes
sinh (x) cosh (x) tanh (x) csch (x) sech (x) coth (x)	Seno hiperbólico (radianes) Coseno hiperbólico Tangente hiperbólico Cosecante hiperbólico Secante hiperbólico Cotangente hiperbólico
a… (x)	Inversa de la función trigonométrica con el resultado expresado en radianes
asin(x) acos (x) atan (x) acsc (x) asec (x) acot (x)	Arco seno (radianes) Arco Coseno Arco Tangente Arco Cosecante Arco Secante Arco Cotangente
a…d(x)	Inversa de la función trigonométrica con el resultado expresado en grados
asind(x) acosd (x) atand (x) acscd (x) asecd (x) acotd (x)	Arco seno (grados) Arco Coseno Arco Tangente Arco Cosecante Arco Secante Arco Cotangente
a…h(x)	Inversa de la función trigonométrica hiperbólica con el resultado expresado en radianes
asinh(x) acosh (x) atanh (x) acsch (x) asech (x) acoth (x)	Arco seno hiperbólico (radianes) Arco Coseno hiperbólico Arco Tangente hiperbólico Arco Cosecante hiperbólico Arco Secante hiperbólico Arco Cotangente hiperbólico

OTRAS OPERACIONES

Es importante saber que **x** e **y** son cualquier escalar, **m** y **n** enteros.

Función	Definición
abs(x)	Valor absoluto o magnitud de un numero complejo

sing(x)	Signo del argumento si x es un valor real (-1 si es negativo, 0 si es cero, 1 si es positivo)
exp(x)	Exponencial
gcd(m,n)	Máximo común divisor
lcm(m,n)	Mínimo común divisor
log(x)	Logaritmo neperiano o natural
log2(x)	Logaritmo en base 2
log10(x)	Logaritmo decimal
mod(x,y)	Modulo después de la división
rem(x,y)	Resto de la división entera
sqrt(x)	Raíz cuadrada
nthroot(x,n)	Raíz n-ésima de x

VECTORES Y MATRICES

Definición:

Para crear un vector se introducen los valores deseados separados por espacios (o comas) todo ello entre corchetes []. Normalmente se usan las letras mayúsculas.

Ejemplo:

>> A= [123;456;789]

```
    1   2   3
A=  4   5   6
    7   8   9
```

Funciones:

Función	Definición
cross (x,y)	Producto vectorial entre los vectores x e y
dot (x,y)	Producto escalar entre los vectores x e y

Si lo que queremos es crear una matriz lo hacemos de forma similar, pero separando las filas con puntos y comas (;). Normalmente se usan las letras minúsculas.

Ejemplo:

>> b= [1 2 3 4 5 6 7 8 9]

B= 1 2 3 4 5 6 7 8 9

Operaciones básicas:

Símbolo	Expresión	Operación
+	A + B	Suma de matrices
-	A - B	Resta de matrices

*	A * B	Multiplicación de matrices
.*	A .* B	Multiplicación elemento a elemento de matrices
/	A / B	División de matrices por la derecha
./	A ./ B	División elemento a elemento de matrices por la derecha
\	A \ B	División de matrices por la izquierda
.\	A .\ B	División elemento a elemento de matrices por la izq.
^	A ^ B	Potenciación (n debe ser un número, no una matriz)
.^	A .^ B	Potenciación elemento a elemento de matrices
'	A ' B	Trasposición compleja conjugada
. '	A . ' B	Trasposición de matrices

Funciones para el análisis de matrices:

Función	Definición
cond (A)	Numero de condición
det (A)	Determinante
diag (V)	Crea una matriz diagonal con el vector v sobre la diagonal
diag (A)	Extrae la diagonal de la matriz A con un vector columna
eig (A)	Valores propios
inv (A)	Matriz inversa
length (A)	Máxima dimensión
norm (A)	Norma
norm (A,n)	Norma-n
normest (A)	Estimación de la norma-2
null (A)	Espacio nulo
orth (A)	Ortogonalizacion
pinv (A)	Pseudoinversa
poly (A)	Polinomio característico
rank (A)	Rango
rref (A)	Reducción mediante la eliminación de Gauss de una matriz
size (A)	Dimensiones
trace (A)	Traza
tril (A)	Matriz triangular inferior a partir de la matriz A
triu (A)	Matriz triangular superior de la matriz A

TEXTO

Una cadena de caracteres es texto rodeado por comillas simples (') y se manejan como vectores filas. Se direccionan y manipulan igual que los vectores. Son posibles las operaciones matemáticas sobre cadenas. Una vez hecha una operación matemática sobre una cadena, ésta se ve como un vector de números en ASCII.

Para ver la representación ASCII de una cadena, podemos utilizar las funciones **abs, double** o sumamos cero. Para restaurarla y verla de nuevo como cadena de

caracteres, usamos la función **setstr**. Si queremos cambiar a minúsculas añadiremos la diferencia entre 'a' y 'A'.

Si queremos que escriba algo en pantalla podemos utilizar el comando **disp**.

HIPERMATRICES

Matlab permite trabajar con matrices de más de dos dimensiones. Los elementos de una hipermatriz pueden ser números, caracteres, estructuras y vectores o matrices de celdas. Las funciones que operan con matrices de más de dos dimensiones son análogas a las funciones vistas anteriormente, aunque con algunas diferencias, por ejemplo, a la hora de definirlas:

```
>> HM(:,:,1) = [1 2 3; 4 5 6];       % definimos la primera capa

>> HM(:,:,2) = [7 8 9; 10 11 12]     % definimos la segunda capa
```

HM(:,:,1) = $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix}$

HM(:,:,2) = $\begin{matrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{matrix}$

Algunas funciones para generar matrices admiten más de dos subíndices y pueden ser utilizadas para generar hipermatrices como **rand, randn, zeros y ones**, también se pueden emplear con hipermatrices las funciones **size** y **reshape** entre otras.

La función **cat** permite concatenar matrices según las distintas “dimensiones”.

Respecto al resto de funciones debemos tener en cuenta que:

1. Las funciones que operan sobre escalares, como sin, cos, etc., se aplican sobre hipermatrices elemento a elemento (igual que ocurre al aplicarlas sobre vectores y matrices).

2. Las funciones que operan sobre vectores, como sum, max, etc., se aplican a matrices e hipermatrices según la primera dimensión, resultando un array de una dimensión inferior.

3. Las funciones matriciales propias del álgebra lineal, como det, inv, etc., no se pueden aplicar a hipermatrices, para aplicarlas habría que extraer las matrices correspondientes.

Para definirlas:

Es una agrupación de datos de tipo diferente bajo un mismo nombre. A los datos les llamamos campos. No hace falta definir previamente el modelo de la estructura, podemos ir creando los distintos campos uno a uno o bien con el comando struct,

donde los nombres de los campos se escriben entre apóstrofos (') seguidos del valor que se les quiere asignar.

Ejemplo:

```
>> alumno.nombre = 'Pablo';  % introduce el campo nombre en la estructura alumno
>> alumno.apellido1 = 'Fernández'; % introduce el campo apellido1 en la estructura alumno
>> alumno.apellido2 = 'García'; % introducimos el campo apellido2 en la estructura alumno
>> alumno.edad = 15; % introducimos el campo edad en la estructura alumno
>> alumno % escribe por pantalla la información almacenada en la estructura

alumno =

nombre: 'Pablo'
apellido1: 'Fernández'
apellido2: 'García'
edad: 15
```

VECTORES Y MATRICES DE CELDAS

Un vector de celdas es un vector cuyos elementos son cada uno de ellos una variable de cualquier tipo. En todo vector sus elementos pueden ser números o cadenas de caracteres, pero en un vector de celdas el primer elemento puede ser un número, el segundo una matriz, el tercero una estructura, etc.

Para crear un vector de celdas usaremos llaves ({}).

OPERADORES RELACIONALES

Operador	Definición
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual a
~=	Distinto de

OPERADORES LÓGICOS

Operador	Definición
&	Y
|	O
~	no

Además de los operadores relacionales y lógicos básicos anteriores, Matlab proporciona una serie de funciones relacionales y lógicas adicionales que incluyen:

Función	Definición
xor (x,y)	Operación “o” exclusiva, devuelve 0 si ambas son falsas o verdaderas y devuelve 1 si una es falsa y la otra verdadera
any (x)	Devuelve 1 si algún elemento en un vector x es no nulo y devuelve 0 si son todos nulos, si se trata de una matriz da una respuesta por cada columna
all (x)	Devuelve 1 si algún elemento en un vector x son no nulos y 0 si existe alguno nulo y se trata de una matriz da una respuesta por cada columna
exist (‘x’)	Devuelve uno si existe y cero si no existe
isnan (x)	Devuelve unos en magnitudes no numéricas (NaN) en x
isinf (x)	Devuelve unos en magnitudes infinitas (Inf) en x
isfinite (x)	Devuelve unos en valores finitos en x

Existe un orden de precedencia para operadores aritméticos, lógicos y relacionales, en la siguiente tabla van de mayor a menor precedencia:

Orden de precedencia de operadores	
1°	^ .^ ' .'
2°	* / \ .* ./ .\
3°	+ - ~ +(unario) -(unario)
4°	: > < >= <= == ~=
5°	| &

GRAFICAS

2-D

La orden **plot** genera una gráfica. Los argumentos deben ser vectores de la misma longitud. La función plot nos permite otras opciones como superponer gráficas sobre los mismos ejes.

Si quieres cambiar la apariencia de la gráfica basta pinchar en el último botón de la barra de herramientas y se abrirán unos cuadros en los laterales que nos permitirán ir haciendo los cambios deseados como darle nombre a los ejes.

Además, podemos colocar etiquetas o manipular la gráfica:

etiqueta sobre el eje X de la gráfica actual: **>> xlabel('texto')**
etiqueta sobre el eje Y de la gráfica actual: **>> ylabel('texto')**
título en la cabecera de la gráfica actual: **>> title('texto')**
texto en el lugar especificado por las coordenadas: **>> text(x,y, 'texto') texto,**
el lugar lo indicamos después con el ratón: **>> gtext('texto')**
dibujar una rejilla: **>> grid**
fija valores máximo y mínimo de los ejes: **>> axis([xmin xmax ymin ymax])**
fija que la escala en los ejes sea igual: **>> axis equal**
fija que la gráfica sea un cuadrado: **>> axis square**
desactiva axis equal y axis square: **>> axis normal**

abre una ventana de gráfico: **>> hold on**
borra lo que hay en la ventana de gráfico: **>> hold off**

Todas estas órdenes se las podemos dar desde la propia ventana de la gráfica una vez que hemos abierto las opciones con el botón indicado anteriormente.

Otros comandos relacionados con las gráficas son los siguientes:

Orden	Definición
area	Colorea el área bajo la gráfica
bar	Diagrama de barras (verticales)
barh	Diagrama de barras (horizontales)
hist	Histograma
pie	Sectores
rose	Histograma polar
stairs	Gráfico de escalera
stem	Secuencia de datos discretos
loglog	Como plot pero con escala logarítmica en ambos ejes
semilogx	Como plot pero escala logarítmica en el eje x
semilogy	Como plot pero escala logarítmica en el eje y

3-D
Gráficos de línea:
También podemos crear gráficas en 3 dimensiones, se trata de extender la orden de plot (2-D) a plot3 (3-D) donde el formato será igual pero los datos estarán en tripletes.

Si queremos representar un polígono en 3 dimensiones lo haremos con la función fill3 de forma similar a **fill** pero ahora con 4 argumentos, siendo el cuarto el que indica el color.

Superficie de malla:

La orden **[X,Y]=meshgrid(x,y)** crea una matriz **X** cuyas filas son copias del vector x y una matriz **Y** cuyas columnas son copias del vector y. Para generar la gráfica de malla se usa la orden **mesh(X,Y,Z)**, **mesh** acepta un argumento opcional para controlar los colores. También puede tomar una matriz simple como argumento: **mesh(Z).**

Gráfica de superficie:

Es similar a la gráfica de malla, pero aquí se rellenan los espacios entre líneas. La orden que usamos es **surf** con los mismos argumentos que para **mesh**.

Las gráficas de contorno en 2-D y 3-D se generan usando respectivamente las funciones **contour** y **contour3**.

La función **pcolor** transforma la altura a un conjunto de colores.

Manipulación de gráficos:
fija el ángulo de visión especificando el azimut y la elevación: **>> view(az,el)**
coloca su vista en un vector de coordenada cartesiana (x,y,z)
en el espacio 3-D: **>> view([x,y,z])**
almacena en **az** y **el** los valores del azimut y
de la elevación de la vista actual: **>> [az,el]=view**
añade etiquetas de altura a los gráficos de contorno: **>> clabel(C,h)**
añade una barra de color vertical mostrando las transformaciones: **>> colorbar**

Comprensión de los mapas de color: (color, nombre corto, rojo/verde/azul)
Negro, K, [0 0 0]
Blanco, w, [1 1 1]
Rojo, r, [1 0 0]
Verde, g, [0 1 0]
Azul, b, [0 0 1]
Amarillo, y, [1 1 0]
Magenta, m, [1 0 1]

PROGRMACIÓN DE MATLAB

Sentencia FOR

Un bloque **for** en cada iteración asigna a la variable la columna i-ésima de la expresión y ejecuta las órdenes. En la práctica las expresiones suelen ser del tipo escalar: escalar en cuyo caso las columnas son escalares.

for variable = expresión
<orden>
<orden>
…
<orden>
end

Sentencia WHILE

Un bloque **while** ejecuta las órdenes mientras todos los elementos de la expresión sean verdaderos.

While <expresión>
<orden>

<orden>
…
<orden>
End

Sentencia IF

Un bloque **if** puede escribirse de varias maneras distintas. Lo que hace es evaluar una expresión lógica y si es cierta ejecuta las órdenes que encuentre antes del **end**.

if <expresión>
<órdenes evaluadas si la expresión es verdadera >
end

Puede que nos interese que en caso de no ejecutar dicha orden ejecute otra distinta. Esto se lo indicaremos usando **else** dentro del bloque.

if <expresión>
<órdenes evaluadas si la expresión es verdadera >
else
<órdenes evaluadas si la expresión es falsa >
end

Si queremos dar una estructura mucho más completa, usaremos la más general donde sólo se evalúan las órdenes asociadas con la primera expresión verdadera de todas. En cuanto la evalúe deja de leer el resto y se dirige directamente al **end**.

If <expresión1>
<órdenes evaluadas si la expresión1 es verdadera >
elseif <expresión2>
<órdenes evaluadas si la expresión2 es verdadera >
elseif<expresión3>
<órdenes evaluadas si la expresión3 es verdadera >
elseif
…
…
else
<órdenes evaluadas si ninguna otra expresión es verdadera >
end

Sentencia BREAK

Si queremos que en un momento dado termine la ejecución de un bucle **for** o un bucle **while** usaremos **break.**

Sentencia CONTINUE

La sentencia **continue** hace que se pase inmediatamente a la siguiente iteración del bucle **for** o del bucle **while** saltando todas las órdenes que hay entre el **continue** y el fin del bucle en la iteración actual.

FUNCIONES EN M-ARCHIVOS

Existen dos tipos de M-archivo, es decir, de archivos con extensión ***.m**. Un tipo son los ficheros de comandos (es un archivo stript) y el otro son las funciones.

Un fichero de comandos contiene simplemente un conjunto de comandos que se ejecutan sucesivamente cuando se teclea el nombre del fichero en la línea de comandos de Matlab o se incluye dicho nombre en otro fichero ***.m**.

Las funciones permiten definir funciones análogas a las de Matlab, con su nombre, argumentos y valores de salida. La primera línea que no sea comentario debe empezar por la palabra **function**, seguida por los valores de salida (entre corchetes **[ ]** y separados por comas si hay más de uno), el signo igual (=) y el nombre de la función seguido de los argumentos (entre paréntesis **( )** y separados por comas):

Function [a,b,c]= nombre_función (x,y,z)

En las líneas siguientes escribimos los argumentos de salida a partir de los de entrada. El nombre de la función y el nombre del archivo deben ser idénticos y no empezar por cifra sino por letra.

Todas las variables dentro de una función se aíslan del espacio de trabajo de Matlab. Las únicas conexiones entre las variables dentro de una función y el espacio de trabajo de Matlab son las variables de entrada y salida.

El número de variables de entrada pasadas a una función está disponible dentro de la función en la variable **nargin** y el número de variables de salida solicitadas cuando una función es llamada, está disponible dentro de la función en la variable **nargout**.

Debemos tener siempre en cuenta que los argumentos pueden ser vectores, luego si queremos que las operaciones se hagan elemento a elemento y no vectorialmente debemos usar el punto.

ANÁLISIS DE DATOS

Matlab ejecuta análisis estadísticos sobre conjuntos de datos. Estos conjuntos de datos se almacenan en matrices orientadas por columnas. También incluye, entre otras, las siguientes funciones estadísticas:

Función	Definición
corrcoef(x)	Coeficientes de correlación
cov (x)	Matriz de covarianzas
cumprod(x)	Producto acumulativo de columnas
cumcum(x)	Suma acumulativa de columnas
diff(x)	Diferencias entre elementos adyacentes de X
hist(x)	Histograma o diagrama de barras
iqr(x)	Rango intercuartílico de la muestra
max(x)	Máximo de cada columna
mean(x)	Media de los valores de vectores y columnas
median(x)	Mediana de los valores de vectores y columnas
min(x)	Mínimo de cada columna
prod(x)	Producto de elementos en columnas
rand(n)	Números aleatorios distribuidos uniformemente
randn(n)	Números aleatorios distribuidos normalmente
range(x)	Rango de cada columna
sort(x)	Ordena columnas en orden ascendente
std(x)	Desviación estándar de la muestra
sum(x)	Suma de elementos en cada columna
tabulate(v)	Tabla de frecuencias del vector
var(x)	Varianza de la muestra

POLINOMIOS

Raíces

Un polinomio se representa por un vector fila con sus coeficientes en orden descendiente, es importante colocar los términos con coeficiente nulo. Así por ejemplo si queremos indicar el polinomio
$5x^4 + 2x^2 - x + 7$ escribiríamos [5 0 2 -1 7].

Para encontrar las raíces de un polinomio **p** se usa la función **roots (p)**. Si conocemos las raíces de un polinomio es posible construir el polinomio asociado mediante la función **poly (r)**.

Matlab trabaja con los polinomios como vectores fila y con las raíces como vectores columna.

Ejemplo:

```
>>p=[1  -9  13  9  -14];     %representa al polinomio x^4 - 9x^3 + 13x^2 - 9x - 14
>>roots (p)                  %calcula sus raíces
ans=
```

7.0000
-1.0000
2.0000
1.0000
>> poly (ans) %devuelve el polinomio generado por esas cuatro raíces
ans=

1.0000 -9.0000 13.0000 9.0000 -14.0000

Otras características

Matlab no tiene incorporada una función para sumar polinomios.

Función	Definición
conv(p,q)	Multiplica los dos polinomios p y q
deconv(c,q)	Divide el polinomio c entre q
polyder (p)	Calcula a derivada del polinomio p
polyder(p,q)	Calcula la derivada del producto de los polinomios p y q
polyval(p,A)	Evalúa el polinomio p en todos los valores de la matriz A

ANÁLISIS NUMÉRICO

Representación gráfica

Existe la función **fplot** que evalúa la función que se desea representar en la gráfica de salida. Como entrada, necesita conocer el nombre de la función como una cadena de caracteres y el rango de representación como un vector de dos elementos: **fplot ('nombre', [valor minm valor max]).**

Otras características:

Función	Definición
diff('f')	Derivada de la función respecto a x
diff('f',t)	Derivada parcial de la función respecto a t
diff('f',n)	Derivada n-ésima de la función respecto a x
feval('f',a)	Evalúa la función en a
fiminbnd('f',a,b)	Calcula el mínimo de una función de una variable
fzero('f',a)	Busca el cero de una función unidimensional f más próxima al punto a
quan('f',a,b)	Aproxima la integral definida (según la cuadratura de Simpson)
trapz(x,y)	Integral numérica trapezoidal de la función formada al emparejar los puntos de los vectores x e y

(f función, n número natural, a y b valores numéricos, x e y vectores del mismo tamaño)

Matlab incorpora una serie de funciones para resolver ecuaciones diferenciales ordinarias. Si se trata de un problema rígido deberíamos usar: **ode15s, ode23s, ode23t u ode23tb**, si por el contrario se trata de un problema sin rigidez: **ode113, ode 23** y **ode45**. Para saber más de estas funciones consultar la ayuda de Matlab.

¿Cómo CONVERTIR UN FICHERO (*.m) EN UN EJECUTABLE(*.exe)?

Si tenemos un fichero ***.m**, lo primero que debemos hacer es asegurarnos de que sea una función, para ello en la primera línea del fichero debe aparecer:

function nombre (el nombre de la función debe coincidir con el nombre del fichero)

Ahora debemos situarnos en el directorio donde tengamos el fichero que queremos transformar usando el comando cd, por ejemplo:

>> cd 'C:\Documents and Settings\Escritorio\Prueba'

Lo que debemos escribir a continuación es el comando **mcc** seguido de **–m** y el nombre del fichero:

>> mcc -m nombre

Con esto nos aparecerá en el mismo directorio donde estamos un ejecutable con el mismo nombre. También aparecerán una carpeta y varios archivos**.**