1 Preliminares

1.1 Consideraciones Generales.

?? MATLAB dieferencia entre mayúsculas y minúsculas.

?? Los nombres de todas las funciones empiezan con minúscula.

?? Colocando un punto y coma al final de una instrucción, elimina la salida de resultados por pantalla.

?? Varias instrucciones se pueden concatenar en la misma línea, separadas por comas o punto y coma. En el primer caso, se muestran por pantalla los resultados obtenidos en la evaluación de cada una.

?? Para continuar una instrucción en la línea siguiente, por ser demasiado larga, se ponen tres puntos suspensivos al final de la primera línea.

1.1.1 Manejo de Teclado

A continuación se presentan un resumen de las teclas a utilizar en la línea de comandos (pantalla de entrada; aparece habitualmente como **EDU>>**):

↑ ó Ctrl + P	Recupera la entrada anterior a la actual
↓ ó Ctrl + N	Recupera la entrada siguiente a la actual
← ó Ctrl + B	Lleva el cursor un carácter a la izquierda
→ ó Ctrl + F	Lleva el cursor un carácter a la derecha
Ctrl + ←	Lleva el cursor una palabra a la izquierda
Ctrl + →	Lleva el cursor una palabra a la derecha
Inicio ó Ctrl + A	Lleva el cursor al comienzo de la línea
Fin ó Ctrl + E	Lleva el cursor al final de la línea actual
Esc	Borra la línea de comandos
Supr. ó Ctrl + D	Borra el carácter indicado por el cursor
⟵	Borra el carácter a la izquierda del cursor
Ctrl + K	Borra toda la línea actual.
‘texto’+ ↑	Recuperamos la última instrucción que comienza con ‘texto’.

1.1.2 Control de operaciones y tiempos

Podemos saber el tiempo utilizado en realizar un conjunto de cálculos, o el número de instrucciones ejecutadas empleando las siguientes instrucciones

flops	Devuelve el número de operaciones en punto flotante que se han ejecutado en una sesión (flops(0) pone el contador de operciones a cero)
cputime	Tiempo de CPU en segundos utilizado por Matlab desde el comienzo de la sesión.
etime	Devuelve el tiempo transcurrido entre dos listas tipo clock (definidas anteriormente)
tic	Activa un contador temporal en segundos que finaliza al utilizar la variable toc
toc	Devuelve el tiempo transcurrido en segundos desde que se activo la variable tic

1.2 Variables

1.2.1 Vectores y Matrices

El MATLAB trabaja habitualmente con valores matriciales, de ahí que su definición y manejo sean fundamentales. Veamos la definición de este tipo de variables y el manejo de sus elementos.

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

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

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

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

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

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

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

variable=[vectorfila1; vectorfila2; ? vector filan] Define una matriz cuyas filas vienen dadas por los valores de los vectores fila, que deben tener la misma longitud.

A continuación mostramos la manera de manejar los elementos de este tipo de variables.

x(n)	Devuelve el n-ésimo elemento del vector x
x([n,m,p])	Devuelve los elementos del vector x situados en las posiciones n-ésima, m-ésima y p-ésima.
x(n:m)	Devuelve los elementos del vector x situados entre el n-ésimo y el m-ésimo, ambos inclusive
x(n:p:m)	Devuelve los elementos del vector x situados entre el n-ésimo y el m-ésimo, ambos inclusive pero separados de p en p unidades
A(m,n)	Devuelve el elemento (m,n) de la matriz A (fila m y columna n)
A([m, n],[p, q])	Devuelve la submatriz de A formada por la intersección de las filas n-ésima y m-ésima y las columnas p-ésima y q-ésima.
A(n:m,p:q)	Devuelve la submatriz de A formada por las filas que hay entre la n-ésima y la m-ésima, y por las columnas que hay entre la p-ésima y la q-ésima
A(a:p:b,c:q:d)	Devuelve la submatriz de A formada por las filas que hay entre la a-ésima y la b-ésima tomándolas de p en p, y por las columnas que hay entre la c-ésima y la d-ésima tomándolas de q en q.
A(:,p:q)	Devuelve la submatriz de A formada por las columnas que hay entre la p-ésima y q-ésima.
A(n:m,:)	Devuelve la submatriz de A formada por las filas que hay entre la n-ésima y la m-ésima
A(n,:)	Devuelve la fila n-ésima de la matriz A
A(:,p)	Devuelve la columna p-ésima de la matriz A
A(:)	Devuelve un vector columna cuyos elementos son las columnas de A situadas por orden
A(:,:)	Devuelve toda la matriz A
[A,B,C]	Devuelve la matriz formada por las submatrices A,B,C,...

1.2.2 Variables simbólicas

Matlab considera simbólica cualquier expresión que se introduzca entre comillas simples. Las variables de estas expresiones son variables simbólicas.

symvar(expr)	Devuelve las variables simbólicas de una expresión
sym(x)	Convierte la variable numérica x a simbólica con representación racional exacta
numeric(x)	Convierte la variable simbólica x a numérica
symrat(x)	Ofrece la representación racional exacta de x

digits(d)	Sitúa la precisión de las variables simbólicas en d dígitos decimales exactos
digits	Da la precisión actual para variables simbólicas
svdvpa(x)	Ofrece el valor de la variable simbólica x con los dígitos de precisión definidos con digits
vpa('expr',n)	Ofrece el resultado de la expresión con n dígitos decimales de precisión
pretty ('expr')	Devuelve la expresión utilizando la escritura matemática habitual.
subs(?expr?,?s?, ?t?)	Sustituye t por s en la expresión.

1.2.3 Aproximaciones y precisión en los cálculos

Matlab representa los resultados con exactitud, pero aunque internamente siempre trabaja con cálculos exactos para no arrastrar errores de redondeo, pueden habilitarse diferentes formatos de representación aproximada, que en ocasiones facilitan la interpretación de los resultados. A continuación se citan los comandos que permiten aproximaciones numéricas.

format long	Ofrece los resultados con 16 cifras decimales
format short	Ofrece los resultados con 4 cifras decimales. Se trata del formato por defecto de Matlab
format long e	Ofrece los resultados con 16 decimales más potencias de 10
format short e	Ofrece los resultados con 4 decimales más potencias de 10
format bank	Ofrece los resultados con 2 cifras decimales
format rat	Ofrece los resultados en forma de número racional aproximado
format +	Ofrece el signo de los resultados (+, - o 0)
format hex	Ofrece los resultados en el sistema hexadecimal
vpa 'operaciones'n	Ofrece el resultado de las operaciones con n dígitos decimales exactos
numeric('expr')	Ofrece el valor de la expresión de forma numérica aproximada según el formato actual activo.
digits(n)	Ofrece los resultados con n dígitos exactos.

1.2.4 Sistemas de Numeración

Matlab permite trabajar con sistemas de numeración de base cualquiera, siempre y cuando se disponga del Toolbox extendido de matemáticas simbólica. Además, permite expresar los números en las diferentes bases. Las funciones de trabajo con sistemas de numeración en diferentes bases que implementa Matlab son las siguientes:

convert(nº_decimal, base, nueva_base) Convierte el número decimal (base 10) especificando a la nueva base especificada.

convert(nº_decimal,binary) Convierte en número decimal especificado a base 2(binario)

convert(nº_decimal,octal) Convierte el número decimal especificado a base 8(octal)

convert(nº_decimal,hex) Convierte el número decimal especificado a base 16 (hexadecimal)

convert(nº_binario,decimal, binary) Convierte el número binario especificado a base decimal

convert(nº_octal,decimal, octal) Convierte el número octal especificado a base decimal

convert(nº_hecadecimal,decimal,hex) Convierte el número base 16 especificado a base decimal

convert([a,b,...,],base,base_antigua,base_nueva Convierte el número cuyas cifras en la base antigua son c...ba, a la base nueva. El resultado es una lista con sus cifras colocadas en el orden inverso al habitual de escritura.

1.2.5 Variable especiales

En MATLAb existen variables de uso común, cuyo valor viene ya preasignado.

pi	3'1415926535897...
i ó j	Unidad imaginaria (raíz cuadrada de-1)
inf	Infinito, por ejemplo 1/0
NaN	Indeterminación (Not a Number, por ejemplo 0/0)
realmin	El menor número real positivo utilizable
realmax	El mayor número real positivo utilizable
eps	Variable permanente cuyo valor es inicialmente la distancia desde 1.0 al siguiente número en coma flotante más elevado. Se trata de la tolerancia por defecto para operaciones en coma flotante (acuracidad relativa en punto flotante). En máquinas actuales (máquinas IEEE) su valor es 2^(-52)
ans	Variable creada automáticamente para representar el último resultado procesado al que no se le ha asignado previamente ninguna variable.

De igual forma, se puede interrogar al sistema sobre sus características o las características de las variables que estamos manejando

finite(x)	Devuelve 1 si x es finito, y cero en otro caso
isinf(x)	Da 1 si x es infinito o – infinito, y cero en otro caso
isnan(x)	Da 1 si x es indeterminado y cero en otro caso
isieee	Da 1 si la máquina es IEEE y da 0 en otro caso
computer	Devuelve el tipo de computador

version	Devuelve la versión actual de Matlab
why	Devuelve un mensaje sucinto
clock	Devuelve una lista con los 6 elementos siguientes [año mes dia hora minutos segundos]
date	Devuelve la fecha del calendario actual
lasterr	Devuelve el último mensaje de error
demo	Ejecuta demostraciones sobre Matlab
pack	Consolida el espacio de trabajo en memoria
diary	Guarda el texto de la sesión actual de Matlab
unix	Ejecuta comandos del sistema operativo Unix
ver	Da información sobre el programa y sus Toolbox
info	Da información acerca de Matlab
subscribe	Da información sobre la subscripción a Matlab
whatsnew	Informa acerca de características nueva de Matlab
hostid	Identifica el número del host servidor

1.3 Operaciones con Matlab

Podemos usar matlab como una computadora numérica de gran potencia. Matlab realiza cálculos exactos. La mayoría de los temas de cálculo numérico son tratados con este software.

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

Símbolo	Operación
+	Suma de escalares, vectores o matrices
-	Resta de escalares, vectores o matrices
*	Producto de escalares o de matrices
.*	Producto de escalares o de vectores
\	A\B=inv(A)*B, siendo A y B matrices
.\	A.\B cociente elemental de B entre A (dim(A)=dim(B))
/	Cociente escalar o B/A=B*inv(A), siendo A y B matrices
./	A./B cociente elemental de A entre B (dim(A)=dim(B))
^	Potencia de escalares o potencia escalar de matriz
.^	Potencia elemental de los elementos de A elevados a los correspondientes

	elementos de B

Al combinar varias operaciones en una misma instrucción, han de tenerse en cuenta los criterios de prioridad habituales entre ellas, que determinan el orden de evaluación de la expresión.

Si queremos que el resultado de una operación aparezca en pantalla con un determinado número de cifras exactas, utilizamos el comando de cálculo simbólico vpa (ver 1.2.2). El resultado de la operación es exacto siempre que aparezca un punto al final del resultado. Si se le pide un nº mas pequeño de cifras exactas que las que realmente tiene el resultado, Matlab redondea por la cifra perdida y completa el resultado con potencias de 1.

1.3.1 Operaciones con racionales.

Los números racionales son cocientes de enteros, y Matlab puede trabajar con ellos con aproximaciones del resultado, como una calculadora, o en modo exacto, de manera que el resultado sea otro racional o entero. Para ello es necesario activar el formato racional con el comando **“format rat”**. Matlab devuelve aproximaciones decimales activando cualquier otro tipo de formato (ver 1.2.3). Una vez habilitado el formato de trabajo racional, las operaciones con racionales serán exactas hasta que no se habilite otro formato distinto. Otra forma de trabajo en modo exacto es usar los comandos correspondientes a operaciones con irracionales o cálculo simbólico.

1.3.2 Operadores lógicos

Cuando deseamos comparar variables, al igual que cuando hemos determinado las características de una variable (ver 1.2.4), se necesitan operadores cuyo resultado sea boleano (cierto o falso). Para ello se cuenta con los operadores lógicos

Operador	Función que desempeña
<	Menor (para complejos sólo afecta a partes reales)
<=	Menor o igual (sólo afecta a partes reales)
>	Mayor (sólo afecta a partes reales)
>=	Mayor o igual (sólo afecta a partes reales)
x==y	Igualdad (afecta a los números complejos)
x? =y	Desigualdad (afecta a los números complejos)

Y con los operadores relacionales

Operador	Función que desempeña

-A	Negociación Lógica (NOT) o complementario de A
A&B	Conjunción lógica (AND) o intersección de A y B
A? B	Disyunción lógica (OR) o unión de A y B
xor(A,B)	OR exclusivo (XOR) o diferencia simétrica de A y B

1.4 Funciones matemáticas.

La librería MATLAB dispone de una gama muy completa de funciones predefinidas que se corresponden con las funciones matemáticas más utilizadas.

1.4.1 Funciones trigonométricas e hiperbólicas

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

1.4.2 Funciones exponenciales

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

1.4.3 Funciones específicas de variable numérica

abs(Z)	Módulo o valor absoluto
angle(Z)	Argumento
ceil(x)	Redondea los decimales al mayor entero más cercano
ceil(Z)	Aplica la función ceil a real (Z) y a imag(Z)
conj(Z)	Complejo conjugado
fix(x)	Elimina la parte decimal del real x
fix(Z)	Aplica la función fix a real (Z) y a imag(Z)
floor(x)	Redondea los decimales al menor entero más cercano

floor(Z)	Aplica la función floor a real (Z) y a imag(Z)
imag(Z)	Parte imaginaria
real(Z)	Parte real
rem(a,b)	Da el resto de la división entre los reales a y b
rem(Z1,Z2)	Resto de la división de los términos de Z1 y Z2
round(x)	El entero más próximo al real x
round(Z)	Aplica la función round a real (Z) y a imag(Z)
sign(x)	Signo del real x (1 si x>0, -1 si x<0)
sign(Z)	Función signo

1.4.4 Funciones Matriciales

expm(Z)	Función exponencial matricial por defecto
expm1(Z)	Función exponencial matricial en M-fichero
expm2(Z)	Función exponencial matricial vía series de Taylos
expm3(Z)	Función exponencial matricial vía autovalores
logm(Z)	Función logarítmica matricial
sqrtm(Z)	Función raíz cuadrada matricial
max(x)	Máximo de los números los elementos de x
min(x)	Mínimo de los números los elementos de x
gcd(x)	Máximo común divisor de los elementos de x
lcm(x)	Mínimo común múltiplo de los elementos de x
maple(`igcd(x)´)	Máximo común divisor de los elementos de x
funm(Z, ‘función’)	Aplica la función a la matriz

1.4.5 Funciones de Léxico

abs(‘cadena_caracteres’)	Devuelve el vector de caracteres ASCII equivalentes a cada carácter de cadena
setstr(vector_numérico)	Devuelve la cadena de caracteres ASCII equivalentes a los elementos del vector
str2mat(t1,t2,te	Forma la matriz cuyas filas son las cadenas de caracteres t1, t2,t3,.., respectivamente
str2num(‘cadena’)	Convierte la cadena de caracteres en su valor numérico exacto utilizado por Matlab
num2str(número)	Convierte el número exacto en su cadena de

	caracteres equivalente con la precisión fijada
int2str(entero)	Convierte en número entero en cadena
sprintf('formato', A)	Convierte la matriz numérica exacta A a una cadena con el formato especificado
sscanf(' cadena' , ' formato', A)	Convierte la cadena a valor numérico con formato especificado
dec2hex(entero)	Convierte el entero decimal a su cadena equivalente en hexadecimal
hex2dec('cadena_hex')	Convierte la cadena hexadecimal en el número entero equivalente
hex2num(`cadena_hex')	Convierte la cadena hexadecimal en el número IEEE en punto flotante equivalente
lower('cadena')	Convierte la cadena a minúsculas
upper('cadena')	Convierte la cadena a mayúsculas
strcmp(c1,c2)	Compara las cadenas s1 y s2 y devuelve 1 si son iguales y 0 en caso contrario
findstr(c, ' exp1' ' exp2')	Reemplaza en la cadena c, exp2 por exp1
findstr(c, 'exp')	Da los lugares que ocupa exp en la cadena c
isstr(expresión)	Da 1 si la expresión es cadena y 0 si no lo es
blanks(n)	Genera una cadena de n blancos
deblank(cadena)	Sustituye los blancos de la cadena por nulos
eval(expresión)	Ejecuta la expresión aunque sea una cadena
disp('cadena')	Muestra la cadena (o matriz) tal y como se ha escrito y continúa el proceso de Matlab
input('cadena')	Muestra la cadena en pantalla y Matlab espera la presión de una tecla para continuar

1.4.6 Números Aleatorios

rand	Devuelve un número decimal aleatorio distribuido uniformemente en el intervalo [0,1]
rand(n)	Devuelve una matriz de dimensión nxn cuyos elementos son números decimales aleatorios distribuidos uniformemente en el intervalo [0,1]
rand(m,n)	Devuelve una matriz de dimensión mxn cuyos elementos son

	números decimales aleatorios distribuidos.
rand(size(A))	Devuelve una matriz del mismo tamaño que la matriz A y cuyos elementos son números decimales aleatorios distribuidos uniformemente en el intervalo [0,1]
rand(\`seed’)	Devuelve el valor actual de la semilla generadora de los números aleatorios uniformes-
rand(\`seed’,n)	Coloca en la cantidad n el valor actual de la semilla generadora de los números aleatorios uniformes
randn	Devuelve un número decimal aleatorio distribuido según una normal de media o y varianza 1
randn(n)	Devuelve una matriz de dimensión nxn cuyos elementos son números decimales aleatorios distribuidos según una normal de media 0 y varianza 1
randn(m,n)	Devuelve una matriz de dimensión mxn cuyos elementos son números decimales aleatorios distribuidos según una normal de media 0 y varianza 1
randn(size(A))	Devuelve una matriz del mismo tamaño que la matriz A y cuyos elementos son números decimales aleatorios distribuidos según una normal de media 0 y varianza 1
randn(\`seed´)	Devuelve el valor actual de la semilla generadora de los númeos aleatorios normales
randn(\`seed´, n)	Coloca en la cantidad n el valor actual de la semilla generadora de los números aleatorios uniformes.

1.4.7 Funciones estadísticas avanzadas

binomial(x,y)	Numero combinado x sobre y: x!/(y!(x-y)!)
bernouilli(n)	Enésimo número de Bernouilli Bn: text/(et-1)=? Bn(x)tn/n! n=0...?
euler(n)	Enésimo número de Euler En: 2/(et- e-t)=? En(x)tn/n! n=0...?
GAMMA(z)	Función Gamma ???: $?(z) ? \int_0^? t^{z?1}e^{?t}dt$
GAMMA (z1,z2)	Función Gamma incompleta $?(a,z) ? \int_0^? t^{a?1}e^{?t}dt$
lnGAMMA(z)	Logaritmo de Gamma: $In?(z) ? In(?(z))$

beta(z1,z2)	Función Beta: $B(z1, z2) ? ?(z1)?(z2)/?(z1 ? z2)$
erf(z)	Función error: $2?/? ?_0^z e^{?t\wedge 2} dt$
erfc(z)	Complemento error: $2?/? ?_0^z e^{?t\wedge 2} dt ? 1 ? erf(z)$
erfc(n,z)	$erfc(n, z) ? ?_0^z erfc(n ? 1, t) dt, erfc(?1, x) ? 2?/? e^{?x\wedge 2} ?$
dawson(x)	Integral de Dawson: $e^{?x\wedge 2} ?_0^x e^{?t\wedge 2} dt$
si(z)	Seno integral: $?_0^z sin(t)/t dt$
ssi(z)	Seno integral desviado= Si(z)-?/2
ci(z)	Coseno integral: $? ? In(iz) ? i?/2 ? ?_0^z (cos(t) ? 1)/t dt$
shi(z)	Seno hiperbólico integral: $?_0^z sinh(t)/t dt$
chi(z)	Cos.hip.integral: $? ? In(iz) ? i?/2 ? ?_0^z (cosh(t) ? 1)/t dt$
s(z)	Seno integral de Fresnel: $?_0^z sin(?/2 t^2) dt$
c(z)	Coseno integral de Fresnel: $?_0^z cos(?/2 t^2) dt$
ei(z)	Integral exponencial : $?_?^x e^{?t}/t dt$ (valor principal)
ei(n,z)	Integral exponencial ampliada: $?_1^? e^{?zt}/t^n dt$
li(x)=Ei(ln(x))	Logaritmo integral: $?_0^x dt/In(t)$ (valor principal)
dilog(x)	Integral dilogarítmica: $?_1^x In(t)/(1 ? t) dt$ x>1
Psi(z)n	Función digamma: $?(z) ? ?'(z)/?(z)$
psi(n,z)	Función Poligamma: $?^n(z) ? d^n/dz^n(?(z))$
harmonic(n)	Función Armónica: $?_{k?1}^n 1/k ? ?(n ? 1) ? ?$
zeta(s)	Zeta de Riemann: $?_{k?1}^n 1/k^s ? ?(s)$

zeta(n,s,q)	$? \; ?_{k?1}^{n} \; 1/(k\ ?\ q)^{s}$
zeta (n,s)	$?\ d^{n} / ds^{n} (?(s))$
hipergeom(n,d,z)	$? \; ?_{K?0}^{?} \; (?_{i?1}^{j} \; (?(n_{i}\ ?\ k)/?(n_{i}))z^{k} / ?_{i?1}^{m} \; (?(d_{i}\ ?\ k)/?(d_{i}))k!)$ n=[n1,n2,..nj] d=[d1,d2,...dm] (Funciºón hipergeométrica generalizada)

1.5 Los ficheros en MATLAB

Todos los ficheros de instrucciones MATLAB deben de llevar la extensión *.m*, denominándoseles por este motivo *m-ficheros*. Dentro de estos ficheros se deben distinguir dos tipos:

?? Ficheros de función: Son aquellos ficheros de instrucciones cuya primera línea ejecutable (no de comentario) comienza con la palabra ***function***. Estos subprogramas de función serán estudiados a continuación.

?? Ficheros de instrucciones: son m-ficheros que no constituyen funciones y que se construyen mediante una secuencia de instrucciones. El contenido de un fichero de programas MATLAB *nombre.m* se ejecuta tecleando simplemente su nombre.

Los ficheros de función y los ficheros de instrucciones tienen las siguientes diferencias:

- ✍✍ El m-fichero de instrucciones sólo puedo ser utilizado a continuación del “prompt” de MATLAB o como una línea en otro m-fichero. Sin embargo, una función puede ser llamada desde cualquier expresión.
- ✍✍ El m-fichero de instrucciones no admite argumentos de entrada, simplemente trabaja con los datos existentes en el espacio de trabajo.
- ✍✍ Las variables que se utilizan para obtener el valor de una función son locales mientras se está ejecutando el m-fichero de la función; en cambio las variables calculadas en un m-fichero de instrucciones son globales y pueden ser utilizadas con posterioridad.
- ✍✍ Aunque dentro de una función modifiquemos el valor de los argumentos de entrada, tal modificación también es local, no transmitiéndose una vez finalizada la ejecución de la función.

1.5.1 Instrucciones de entrada y salida.

Todos los m-ficheros admiten ordenes que permiten mantener una comunicación con el usuario sobre la ventana de instrucciones. Entre las más destacadas están:

x = ***input*** (´mensaje´[,'s']) Permite la introducción de datos por pantalla. La opción *'s'* se emplea para leer una variable de tipo carácter ('string'), evitando los apóstrofes.

disp*(´mensaje´) ó disp(´texto´)* Muestra un texto o una matriz de texto por pantalla. Para combinar información numérica y texto en un comando disp se deben utilizar las instrucciones int2str, num2str y mat2str.

x = ***menu*** *('título', 'opción_a' [,'opción_b', ..., 'opción_k'])* Genera un menú que permite al usuario elegir entre distintas opciones.

error*(´mensaje´)* Envía un mensaje a pantalla, informando al usuario que ha ocurrido un error y detiene la ejecución del programa, devolviendo el control al teclado.

echo *on/off* Activa o desactiva la escritura de cada instrucción del fichero sobre la pantalla

pause *(10)* Detiene la ejecución del fichero hasta que se pulse alguna tecla o transcurre el nº indicado de segundos

keyboard Detiene la ejecución de un fichero y permite al usuario intercalar una serie de instrucciones. La ejecución continuará cuando hagamos **return** desde la ventana de instrucciones

1.5.2 Lectura y escritura en ficheros externos

MATLAB permite salvar y recuperar datos de diferentes tipos de ficheros, diferenciándolos básicamente por su extensión.

mat Fichero de datos binario. Se genera automáticamente con la instrucción **save** *fichero* y se recupera mediante la instrucción **load**. Hay que indicar que se pueden salvar los datos como caracteres ASCII empleando la opción *–ascii*, en cuyo caso el fichero no toma la extensión *.mat*.

dia Fichero ascii que almacena las instrucciones ejecutadas durante una sesión de trabajo. Se genera automáticamente mediante la instrucción **diary**, pudiendo posteriormente convertirse en un *m-fichero*.

dat Fichero de datos, generados por el usuario. Si son homogéneos (no se mezclan distintos tipos de datos) y mantienen la estructura (mismo numero de datos por linea) pueden recuperarse mediante la instrucción **load**.

1.5.2.1 Guardando y leyendo datos en ficheros de formato Matlab y ASCII

El comando save es el instrumento esencial para guardar datos en ficheros tipo matlab. Su recíproca es la instrucción load.

***save** file var opciones* Almacena todas las variables indicadas en el fichero file de formato matlab binario o ASCII dependiendo de las opciones.

***load** file* Recupera todas las variables del fichero file.

Entre las opciones se encuentran

-ascii Salva los valores en formato ASCII de 8 dígitos.

-double Salva los valores en formato ASCII de 16 dígitos.

-tabs Separa los valores por tabuladores (sólo con las opciones anteriores).

Las distintas modalidades de uso de ambos comandos se presentan a continuación.

save	Almacena todas las variables del espacio de trabajo en el fichero de formato matlab binario "matlab.mat".
save X Y	Idem, pero almacenando sólo las variables X e Y.
save file	Idem, pero usando el fichero "file.mat"
save -ascii	Almacena todas las variables del espacio de trabajo en el fichero de formato matlab "matlab".
save -ascii-double	Idem en formato ASCII de 16 dígitos.
save –ascii -tabs	Idem en formato ASCII de 8 dígitos con valores delimitados por tabuladores
load	Lee todas las variables guardadas con el comando save en el fichero de nombre matlab.mat
load file	Lee las variables del fichero binario file.mat
load file.txt	Lee el fichero ASCII de nombre file.txt

1.5.2.2 Guardando y leyendo datos en ficheros cualesquiera

Siempre que se quieran leer o escribir datos en un fichero cualquiera (que no tiene porqué ser de formatos ASCII o Matlab), será necesario utilizar en primer lugar el comando fopen para abrirlo. Después se usarán los comandos correspondientes de lectura y escritura (fload, fwrite, fscanf, fprintf, etc) con el fin de realizar las correspondientes operaciones de lectura o escritura en él. Por último, se utiliza el comando fclose para cerrar el fichero.

El fichero que se abre puede ser nuevo o puede existir previamente (con la finalidad de ampliar su contenido o simplemente leerlo.) El comando fopen devuelve un identificador de fichero que consiste en un entero no negativo asignado por el sistema operativo al fichero que se abre. El indicador es realmente una referencia para el manejo del fichero abierto que posteriormente será utilizada para leerlo (comando read), escribir en él correctamente, fopen devuelve -1 como identificador de fichero. Como identificador de fichero genérico suele utilizarse fid. A continuación se muestra la sintaxis de las diferentes instrucciones y algunos ejemplos

*fid=**fopen**(´file´,'permiso','formato')* Abre el fichero file y le asigna un número de identificación que se guarda en la variable fid. A partir de su ejecución, cualquier referencia a ese fichero se hará con fid. Si no se ha podido abrir el archivo, fid toma el valor –1.

ok **= *fclose* (***fid***)** Cierra el archivo identificado con *fid* o todos (**all**). La variable ok toma el –1 cuando la instrucción no ha sido completada correctamente y 0 en otro caso.

Se pueden dar permisos para leer, escribir, añadir, etc. Asimismo, el formato indica si es el nativo de la máquina, IEEE, Vax, Cray, etc.

fid=fopen('pepe','a+','n') Abre el fichero 'pepe' para leer y añadir (ya debe estar creado), utilizando el formato nativo de la máquina.

[filename,permission,architecture]=fopen(fid) Devuelve el nombre del fichero, el tipo de permiso y el formato numérico de la arquitectura especificada referente al fichero cuyo identificador es fid

n **= *fprint/fwrite* (***fid***,***´formato´***,** *a***)** Escribe los valores de la variable a en el fichero ASCII o binario respectivamente. Sus opciones son análogas a las instrucciones de lectura.

[a ,n] = ***fscanf/fread*** *(fid, ´formato´, m)* Lee datos del archivo ASCII o binario respectivamente, identificado por *fid*. Los *n* datos leídos correctamente se guardan en la variable *a*. La variable *m* indica el número de datos que se desean leer; si *m* es una matriz, se leerán tantos datos como elementos tenga, rellenándose por columnas. Para leerlos todos *m=inf*.

El argumento de formato consiste en una cadena formada por caracteres de escape (precedidos del carácter "\") y por caracteres de conversión según los diferentes formatos (precedidos del carácter "%"). Los posibles caracteres de escape y de conversión son respectivamente:

\n Se ejecuta el paso a nueva línea

\t Se ejecuta un tabulador horizontal

\b se ejecuta un paso hacia atrás de un solo carácter (backspace), borrando el contenido del carácter actual en caso de que exista

\r Se ejecuta un retorno de carro

\f Se ejecuta un salto de página (form feed)

\\ Se ejecuta el carácter backslash

\' Se ejecuta el carácter comilla simple

%d Enteros en el sistema decimal

%o Enteros en el sistema octal

%x Enteros en el sistema hexadecimal

%u Enteros sin signo en el sistema decimal

%f Reales de punto fijo

%e Reales de punto flotante

%g Utiliza d, e o f, seleccionando el de mayor precisión en el mínimo espacio

%c Caracteres individuales

%s Cadena de caracteres

ok **= frewind(*fid*)** Coloca el puntero al inicio del archivo *fid*.

ok = fseek (fid, n, origen) Coloca el puntero del archivo identificado con *fid* en la posición indicada por la variable *n* (si *n*>0 se avanza el puntero, en caso contrario se retrasa). La variable carácter *origen* indica desde donde se empieza a mover el puntero, tomando los valores: **bof** (inicio del fichero), **cof** (posición actual) o **eof** (final del archivo).

ok = ftell (fid) Indica el número de bytes, contados desde el principio del archivo, hasta la posición donde se encuentra el puntero.

1.6 Programación en MATLAB

Al igual que en los lenguajes de alto nivel, MATLAB permite crear programas utilizando programación estructurada. Asimismo utiliza muchos de los recursos de la programación orientada a objetos.

1.6.1 Estructuras de control condicionadas

Permite seleccionar entre dos conjuntos alternativos de instrucciones dependiendo de que se verifique una condición lógica (cuyo resultado es cierto o falso). Su sintaxis es de la forma:

if *condición*

Instrucciones que deben ejecutarse si la condición 1 es cierta.

else

Instrucciones a ejecutar si no se verifica la condición anterior

end

Cuando no hay instrucciones que ejecutar si la condición no se cumple, la sintaxis anterior se reduce a **if** … **end**. Al contrario, cuando se encadenan varios bloques alternativos, la sintaxis queda como:

if *condición_1*

Instrucciones a ejecutar cuando se verifica la condición 1

elseif *condición_2*

Instrucciones a ejecutar cuando no se verifica la condición 1 y sí la condición_2

elseif *condición_3*

Instrucciones a ejecutar cuando no se verifican las condiciones anteriores y sí la condición_3

…

else

Instrucciones a ejecutar cuando no se verifican las condiciones anteriores

end

Podemos imponer más de una condición o condiciones complejas utilizando los operadores relacionales (condiciones cuyo resultado es cierto o falso) combinados con operadores lógicos (sirven como nexo entre varios relacionales). Entre los principales operadores relacionales están *menor* (<), *menor o igual* (<=), *mayor* (>), *mayor o igual* (>=) e *igual* (==). Entre los operadores lógicos cabe destacar *y* (&), *o*(|) y la *negación* (~). Otros operadores son el *o exclusivo* (**xor**), el *existe alguno* (**any**) y el *todos* (**all**). Los dos últimos se suelen aplicar a vectores, devolviendo *1* (verdadero) cuando algún elemento comple la condición o cuando la cumplen todos los elementos respectivamente.

Otras funciones útiles con resultados lógicos son **insempty** (cierto si el vector está vacío), **isequal** (cierto si las matrices son idénticas), **isreal, insnan, isfinite, isinf,** etc. La función **find** indica la posición de los elementos de un vector que cumplen determinadas condiciones.

1.6.2 Otras estructuras de control

Permite seleccionar conjuntos alternativos de instrucciones dependiendo del valor de un argumento común. Dicho argumento debe ser un escalar o una cadena de caracteres. Su sintaxis es:

switch *argumento*

case *conjunto_1*

Instrucciones a ejecutar cuando *argumento* ? *conjunto_1*

case *conjunto_2*

Instrucciones a ejecutar cuando *argumento* ? *conjunto_2*

case *conjunto_3*

Instrucciones a ejecutar cuando *argumento* ? *conjunto_3*

…

otherwise

Instrucciones a ejecutar cuando *argumento* no pertenece a ningún conjunto previo

end

1.6.3 Bucles simples

Permite repetir un número determinado de veces un conjunto de instrucciones. Su sintaxis es la siguiente:

for var = *vector*

Instrucciones que deben ejecutarse

end

El argumento *vector* puede ser efectivamente un vector, en cuyo caso la variable va tomando los valores de las componentes del vector, o una estructura de la forma *inicio : incremento : fin*, en cuyo caso la variable va tomando valores desde *inicio* hasta *fin* con un determinado *incremento.* Si no se indica el valor del incremento, este se toma como unidad. El número de veces que se repite el bucle viene dado por la dimensión del vector o por la expresión $max\left\{0, \frac{fin - inicio}{incremento} + 1\right\}$.

La ejecución del bucle puede interrumpirse en cualquier momento mediante la instrucción **break**.

1.6.4 Bucles condicionales

Permite repetir un conjunto de instrucciones, en tanto se satisfaga una condición lógica. Su sintaxis es la siguiente:

while *condición*

Instrucciones que deben ejecutarse mientras la condición sea cierta.

end

1.6.5 Funciones

Una función se define mediante un m-fichero, cuyo nombre coincide con el de la función. La primera línea ejecutable la palabra **function**. Su sintaxis es

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

seguida de las instrucciones necesarias. Cuando hay más de un argumento de salida, éstos deben ir entre corchetes y separados por comas. Es conveniente utilizar las primeras líneas del fichero comentario (iniciandolas con '%'), explicando cómo debe usarse la función y sus argumentos (tanto de entrada como de salida). Así, dicha definición será visible mediante la instrucción **help** *nombre-función.*

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

Las variables definidas en la función (salvo los argumentos) son locales. Para que el valor de una variable sea compartido por varias funciones se emplea la instrucción **global**, cuya sintaxis es **global** *variable*, y debe aparecer en todas las funciones que la compartan

Una función utiliza las siguientes instrucciones para verificar el número de argumentos:

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

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

nargchk verifica que el número de argumentos de entrada calculados con **nargin** es válido, devolviendo en caso contrario un mensaje de error.

Para la evaluación de una función también puede utilizarse la instrucción *feval*.

feval ('file',arg1,arg1,..,argn) Evalúa la función file, almacenada en file.m, con los valores de los argumentos arg1,arg2,...,argn

Para definir funciones de una sóla variable se puede utilizar la definición simbólica mediante

nombre= 'función'

Para hallar el valor de la función *nombre* en un punto *a* se utiliza el comando **subs**, cuya sintaxis es la siguiente:

subs(f,a)	Aplica la función f en el punto a
subs(f,a,b)	Sustituye en f el valor b por el valor a

1.6.6 Instrucciones al sistema operativo.

Por último se indican algunas instrucciones MATLAB que nos permiten "comunicarnos" con el sistema operativo.

!	Abandona la ventana de MATLAB y abre una ventana para trabajar con el Sistema Operativo (S.O.)
!instrucción	Ejecuta la instrucción del S.O. desde MATLAB
what	Muestra todos los m-ficheros del directorio actual
dir ó ls	Muestra todos los ficheros del directorio actual
type file	Muestra el contenido del fichero file en la ventana de instrucciones. Si no declaramos la extensión del fichero asumirá file.m,.
delete file	Borra el archivo fichero. Si no declaramos la extensión del fichero borra file.m.
cd ó chdir ó pwd	Muestra el directorio actual de trabajo
cd path	Cambia al directorio path
which file	Informa del directorio (con el path) en el que se encuentra el fichero file. Si no declaramos la extensión del fichero busca file.m.

1.6.7 La orden Help

La orden Help sirve para obtener ayuda sobre un tema conocido. Escribiendo help y a continuación la orden sobre la que queremos obtener información, por ejemplo help sqrt, aparece en la pantalla la información sobre esta orden.

Si no sabemos el tema exacto sobre el que queremos ayuda escribiendo únicamente help obtenemos una guía en la que aparecen las distintas categorías sobre las cuales podemos pedir ayuda.

Como una alternativa para obtener ayuda podemos utilizar la opción Help del menú principal. Llevando el puntero del ratón sobre la palabra Help que aparece en la barra situada en la parte superior de la ventana (Barra de menú) y pulsando el botón izquierdo del ratón ('clic') se obtiene un menú desplegable en el que aparecen cuatro subopciones.

Haciendo `clic` sobre la subopción Index.. obtenemos una lista ordenada alfabéticamente de las órdenes de MATLAB. Si elegimos ('clic') una función determinada de la lista, por ejemplo ABS, se obtiene una pantalla con toda la ayuda referente a esa función.

Todavía podemos obtener esta información de otra manera que consiste en escribir la orden en la ventana de trabajo, a continuación se arrastra el ratón con el botón izquierdo pulsado desde el inicio hasta el final de la palabra, de forma que al soltar el botón izquierdo del ratón la palabra queda escrita en blanco sobre fondo azul. Hacemos 'clic' sobre Help y elegimos la subopción Help Selected. El resultado es el mismo que el de la operación anterior.

1.6.7.1 HELP topics

Toolbox\local	Librería de funciones locales
matlab\datafun	Análisis de datos y transformada de Fourier
matlab\elfun	Funciones matemáticas elementales
matlab\elmat	Matrices elementales y manupulación de matrices
matlab\funfun	Funciones de métodos numéricos no lineales
matlab\general	Instrucciones de propósito general
matlab\color	Funciones de iluminación y control de color
matlab\graphics	Funciones gráficas de propósito general
matlab\iofun	Funciones de I/O de bajo nuvel
matlab\lang	Construcción y depuración del lenguaje
matlab\matfun	Funciones matriciales y Álgebra Lineal numérica
matlab\ops	Operadores y caracteres especiales
matlab\plotxy	Gráficos en dos dimensiones
matlab\plotxyz	Gráficos en tres dimensiones
matlab\polyfun	Polinomios e interpolación de funciones
matlab\sounds	Funciones de tratamiento de sonidos
matlab\sparfun	Funciones de matrices dispersas
matlab\specfun	Funciones matemáticas especiales
matlab\specmat	Matrices especiales
matlab\strfun	Funciones de cadenas de caracteres
matlab\dde	Librería DDE
matlab\demos	Demostraciones de MATLAB

toolbox\symbolic Librería de Matemática Simbólica

1.7 Gráficos

Matlab produce gráficos de dos y tres dimensiones, así como contornos y gráficos de densidad. Se pueden representar los gráficos y listar los datos, permite el control de colores, sombreados y otras características de los gráficos, también soporta gráficos animados. Los gráficos producidos por Matlab son portables a otros programas.

1.7.1 Gráficos bidimensionales (2-D)

Las instrucciones básicas que utiliza Matlab para dibujar la gráfica de una función de una variable son los siguientes:

plot(X) Representa los puntos (k,X_k). Si X es una matriz, hace lo mismo para cada columna de la matriz. Si X es un vector complejo, representa Real(X) frente a IMAG(X)..

plot(X,Y) Representa el conjunto de puntos (X,Y). Si X o Y son matrices, representa por filas o columnas los datos de X frente a los datos de Y, dependiendo si el otro vector es fila o columna. Para valores complejos de X e Y, se ignoran las partes imaginaria.

plot(X,Y,S) Gráfica de plot(X,Y) con la opciones definidas en S. Usualmente, S se compone de dos caracteres entre comillas simples, el primero de los cuales fija el color de la línea del gráfico, y el segundo fija el carácter a usar en el graficado. Los valores posibles de colores y caracteres son, respectivamente, los siguientes:

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

plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3) Combina, sobre los mismos ejes, los gráficos definidos para las tripletas (Xi, Yi. Si). Se trata de una forma de representar varias funciones sobre el mismo gráfico.

fplot('función',[xmin,xmax]) Grafica la función en el intervalo de variación de x dado.

fplot('función',[xmin,xmax, ymin,ymax],S) Gráfica la función en los intervalos de variación de x e y dados, con las opciones de color y caracteres dadas por S.

fplot([f1,f2,..fn],[xmin,xmax, ymin,ymax],S) Gráfica las funciones f1,f2,..fn sobre los mismo ejes en los intervalos de variación de x e y especificados, y con las opciones de color y caracteres dadas por S.

ezplot('funcion',[xmin xmax]) Gráfica la función en el intervalo de variación de x dado

1.7.2 Titulos, etiquetas, mallas y textos

title('texto')	Añade el texto como título del gráfico en la parte superior del mismo en gráficos 2-D y 3-D
xlabel('texto')	Sitúa el texto al lado del eje x en gráfico 2-D y 3-D
ylabel('texto')	Sitúa el texto al lado del eje y en gráficos 2-D y 3-D
zlabel('texto')	Sitúa el texto al lado de eje z en un gráfico 3-D
text('x,y,texto')	Sitúa el texto en el punto (x,y) dentro del gráfico 2-D
text('x,y,z,texto')	Sitúa el texto en el punto (x,y,z) en el gráfico 3-D
gtex('texto')	Permite situar el texto en un punto seleccionado con el ratón dentro de un gráfico 2-D
grid	Sitúa rejillas en los ejes de un gráfico 2-D o ·3-D. La opción grid on coloca las rejillas y greed off las elimina. La opción grid permuta entre on y off
hold	Permite mantener el gráfico existente con todas sus propiedades, de modo que el siguiente gráfico que se realice se sitúe sobre los mismos ejes y se superponga al existente. La opción hold on activa la opción y hold off la elimina. La opción hold permuta entre on y off. Válido para 2-D y 3-D

A continuación se presentan comandos que permiten manipular los ejes de un gráfico, la colocación del mismo dentro de la pantalla, su apariencia, su presentación desde distintos puntos de vista, etc.

axis([xmin xmax ymin ymax])	Sitúa los valores máximo y mínimo para los ejes X e Y en el gráfico corriente
axis ('auto')	Sitúa los ejes en la escala automática por defecto (la dada por xmin=min(x), xmax=max(x) e y libre)
axis (axis)	Congela el escalado de ejes en los límites corrientes, de tal forma que al situar otro gráfico sobre los mismo ejes (con hold en on), la escala no cambie
V=axis	Da el vector V de 4 elementos, conteniendo la escala de gráfico corriente
axis('ij')	Sitúa coordenadas con el origen en la parte superior izquierda del gráfico
axis('square')	Convierte el rectángulo de graficado en un cuadrado, con lo que las figuras se abomban
axis('equal')	Sitúa el mismo factor de escala para ambos ejes
axis ('normal')	Elimina las opciones square y equal
axis('off')	Elimina las etiquetas y marcas de los ejes y las rejillas, manteniendo el título del gráfico y los textos situados en él con text y gtext
axis('on')	Coloca de nuevo las etiquetas, marcas y rejillas de los ejes
subplot(m,n,p)	Divide la ventana gráfica en mxn subventanas y coloca el gráfico corriente en la ventana p-ésima, empezando a contar por la parte superior izquierda y de izquierda a derecha hasta acabar la línea, para pasar a la siguiente
ginput(n)	Permite recuperar las coordenadas de n puntos de un grafico mediante ratón o teclado

1.7.3 Gráficos logarítmicos y semilogarítmicos

Los comandos que habilita Matlab para representar gráfico con escalas logarítmicas son los siguientes

loglog(X,Y)	Realiza los mismos gráficos que plot(X,Y), pero con escala logarítmica en los dos ejes. El comando loglog presenta las mismas variantes y admite las mismas opciones que el comando plot
semilogx(X,Y)	Realiza los mismos gráficos que plot(X,Y) , pero con escala logarítmica en el eje x, y escala normal en el eje y (escala semilogarítmica
semilogy(X,Y)	Realiza los mismos gráficos que plot(X,Y), pero con escala logarítmica en el eje y, y escala normal en el eje x (escala semilogarítmica)

1.7.4 Gráficos en coordenadas polares

Matlab habilita el comando específico polar , que representa funciones en coordenadas polares. Su sintaxis es la siguiente:

polar (?,r)	Representa la curva en coordenadas polares r=r(?)
polar (?,r,S)	Representa la curva en coordenadas polares r=r(?) con el estilo de línea dado por S, cuyos posibles valores ya fueron especificados en el comando plot

1.7.5 Gráfico tridimensionales (3-D)

Los comandos básicos que utiliza Matlab para dibujar gráficos que generan una línea en tres dimensiones son los siguientes:

plot3(X,Y,Z)	Representa el conjunto de puntos (X,Y,Z), donde X,Y y Z son vectores fila. X.Y y Z pueden ser matrices de la misma dimensión, en cuyo caso se hace una gráfica por cada tripleta de filas y sobre los mismos ejes. Para valores complejos de X, Y y Z se ignoran las partes imaginarias.
plot3(X,Y,Z,S)	Gráfica de plot(X,Y,Z) con la opciones definidas en S. Usualmente, S se compone de dos caracteres entre comillas simples, el primero de los cuales fija el color de la línea del gráfico, y el segundo fija el carácter a usar en el graficado. Los valores posibles de colores y caracteres se han descrito ya al explicar el comando plot

plot3(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3) Combina, sobre los mismos ejes, los gráficos definidos para las tuplas (Xi, Yi. Zi, Si). Se trata de una forma de representar varias funciones sobre el mismo gráfico.

1.7.6 Histogramas

bar(Y)	Dibuja el gráfico de barras relativo al vector Y
bar(X,Y)	Dibuja el gráfico de barrar relativo al vector Y cuyos elementos son especificados a través del vector X
stairs(Y)	Dibuja el gráfico escalonado relativo al vector Y
stairs(X,Y)	Dibuja el gráfico escalonado relativo al vector Y cuyos elementos son especificados a través del vector x

del vector x.

hist(Y) Dibuja el histograma relativo al vector Y utilizando 10 rectángulos verticales de igual base

hist(Y,n) Dibuja el histograma relativo al vector Y utilizando n rectángulos verticales de igual base

hist(X,Y) Dibuja el histograma relativo al vector Y utilizando rectángulos verticales cuyas bases miden lo especificado en los elementos del vector X

errorbar(x,y,e) Realiza el gráfico del vector x contra el vector y con los errores especificados en el vector e. Pasando por cada punto (xi,yi)

stem(Y) Dibuja el gráfico de racimo relativo al vector Y. Cada punto del vector Y es unido al eje x por una línea vertical

stem(X,Y) Dibuja el gráfico de racimo relativo al vector Y cuyos elementos son especificados a través del vector X

rose (Y) Dibuja el histograma angular relativo al vector Y de ángulos en radianes, utilizando 20 radios iguales

rose (Y,n) Dibuja el histograma angular relativo al vector Y utilizando n radios iguales.

rose (X,Y)) Dibuja el histograma angular relativo al vector Y utilizando radios que miden los especificado en los elementos del vector X

Compas(Z) Realiza un diagrama de flechas que salen del origen y cuya magnitud y dirección vienen determinadas por el módulo y el argumento de los números complejos componentes del vector Z. La fecha relativa al complejo Zi une el origen con el afijo de Zi

compas(X,Y) Equivale a compas (X+i*Y)

compas(Z,S) o compas(X,Y,S) Especifica en S el tipo de línea a usar en las flechas

feather(Z) o feather(X,Y) o feather(Z,S) o feather (X,Y,S) Es igual que compas, con la única diferencia de que el origen de las flechas no está en el origen de coordenadas, sino que salen de puntos igualmente espaciados de una línea horizontal

legend('leyenda1''leyenda2',..., 'leyendan' Sitúa las leyendas dadas en n gráficos consecutivos

1.7.7 Gráficos de mallas y de superficies

Un gráfico de malla tridimensional de malla viene definido por una función z=f(x,y), de tal forma que los puntos de la superficie se representan sobre una rejilla, resultado de levantar los valores de z dados por f(x,y) sobre los correspondientes puntos del plano (x,y). El aspecto de un gráfico de malla es como una red de pesca, con los puntos de la superficie sobre los nudos de la red. Realmente, es un gráfico de superficie cuyo grafo tiene forma de red. Para representar un gráfico de malla, se utiliza el comando mesh y sus variantes, cuya sintaxis es la siguiente:

mesh(X,Y,Z,C) Representa el gráfico de malla de la función z=f(x,y), dibujando las líneas de la rejilla que componen la malla con los colores especificados en C. El argumento C se puede ignorar.

meshz(X,Y,Z,C) Representa el gráfico de malla de la función z=f(x,y) con una especie de cortina o telón en la parte inferior

	parte inferior
meshc(X,Y,Z,C)	Representa el gráfico de malla de la función z=f(x,y) junto con el gráfico de contorno correspondiente (curvas de nivel proyectadas sobre el plano XY)

Los gráficos de superficie permiten obtener representaciones densas de figuras tridimensionales, y en especial de funciones de dos variables. El primer paso para representar una función de dos variables z= f(x,y) mediante su gráfico de superficie, es utilizar el comando meshgrid, que básicamente define la matriz< de puntos (X,Y) sobre los cuales se evalúa la función de dos variables para hacer su presentación gráfica. Su sintaxis es la siguiente:

[X,Y]=meshgrid(x,y)	Transforma el campo de definición dado de las variables x e y de la función a representar z=f(x,y) en argumento matriciales utilizables por el comando mesh para obtener el gráfico de malla
surf(X,Y,Z,C)	Representa el gráfico de superficie de la función z=f(x,y), realizando el dibujo con los colores especificados en C. El argumento C se puede ignorar
surfl(X,Y,Z)	Representa el gráfico de superficie de la función z=f(x,y), realizando el dibujo con sombreado.

Veamos por último la sintaxis de los gráficos de contornoo curvas de nivel

contour(Z)	Dibuja el gráfico de contorno (curvas de nivel) para la matriz Z. El número de líneas de contorno a utilizar se elige automáticamente
contour(Z,n)	Dibuja el gráfico de contorno (curvas de nivel) para la matriz Z usando n líneas de contorno
contour(z,y,Z,n)	Dibuja el gráfico de contorno (curvas de nivel) para la matriz Z usando en los ejes X e Y el escalado definido por los vectores x e y (n líneas de contorno)
contour3(Z), contour3(Z,n) y contour3(x,y,Z,n)	Dibujan los gráficos de contorno en 3 dimensiones
pcolor(X,Y,Z)	Dibuja un gráfico de contorno (curvas de nivel) para la matriz (X,Y,Z) utilizando una representación basada en densidades de colores. Suele denominarse gráfico de densidad

2 Ecuaciones No Lineales

Normalmente, las funciones definidas en Matlab operan sobre sus argumentos. Sin embargo, también existen operadores funcionales que operan sobre otras funciones (los argumentos de los operadores funcionales son funciones), como, por ejemplo, el operador función inversa. A su vez, Matlab también permite las operaciones clásicas entre funciones(suma, producto, etc.)

symop(expr)	Permite crear nuevas expresiones simbólicas a partir de variables, expresiones y operadores simbólicos; acepta dieciséis argumentos separados por comas, cada uno de los cuales puede ser una expresión simbólica, un valor numérico o un operador aritmético.
symadd(f,g,...) symop(f,'+',g,...)	Realiza la suma las funciones f+g+...
symsub(f,g,...) symop(f,'-',g,...)	Realiza la diferencia de las funciones f –g -...
symmul(f,g) symop(f,'*',g,...)	Realiza el producto delas funciones f*g*...
symdiv(f,g,...) symop(f,'/,'g...)	Realiza el cociente de las funciones f / g / ...
sympow(f,g) symop(f,'^',g)	Eleva f a la potencia g
compose(f,g)	Permite crear la función simbólica compuesta de f y g (ambas definidas como funciones del mismo argumento).
compose(f,g,'u')	Función compuesta de f y g, tomando la expresión u como dominio de f y g
finverse(f)	Devuelve la función inversa de f, avisando cuando el resultado no es único

2.1 Ecuaciones Algebraicas y Trascendentes

Entre los operadores funcionales y operaciones típicas clásicas entre funciones que habilita Matlab podemos resaltar los siguientes:

solve(?expr?,?variable?)	Resuelve la igualdad; si no se especifica la variable toma por defecto la variable simbólica x.
expand (?expr?)	Expande lo más posible una expresión algebraica, realizando totalmente los productos y potencias, hasta presentar el resultado como una suma de términos. Aplica reglas de ángulos múltiples para expresiones trigonométricas y aplica formalmente las propiedades de las funciones logarítmicas y exponenciales. También descompone fracciones algebraicas de numerador polinómico en sumas de

	fracciones
factor (?expr?)	Escribe una expresión algebraica expandida como producto de factores (inversa de la anterior). La factorización se realiza por defecto en el cuerpo definido por los coeficientes de la expresión. para fracciones algebraicas, factoriza numerado y denominador y simplifica factores comunes.
factor (‘expr,a’)	Factoriza la expresión algebraica polinómica en la extensión algebraica del cuerpo de sus coeficientes definida por a (a suele ser un radical simple o compuesto, o una expresión RootOff
simplify(?expr?)	Simplifica lo más posible una expresión algebraica, realizando sumas finales. Ejecuta sumas de fracciones algebraicas, pero no las simplifica totalmente
Simplify (‘expr, regla1, regla2,..reglan’)	Simplifica la expresión teniendo en cuenta las reglas especificadas. Los valores posibles de las reglas son Ei. GAMMA, atsign, hypergeom, In, polar, power, radical, sqrt y trig, y permite simplificaciones de expresiones que contienen Integrales exponenciales, funciones gamma, operadores funcionales, funciones hipergeométricas, logaritmos, funciones polares, radicales, raíces cuadradas y funciones trigonométricas, respectivamente
simplify(‘expr, assume=propiedad’)	Simplifica la expresión teniendo en cuenta la propiedad especificada
simplify(‘expr, symbolic’)	Simplifica la expresión considerando positivas todas las subexpresiones afectadas por radicales
simple (?expr?)	Halla la forma más simplificada posible de la expresión algebraica. es el comando más eficiente para simplificar totalmente una fracción algebraica
collect(‘expr’, ‘x’)	Agrupa la expresión algebraica polinómica en potencias ordenadas de la variable x. si no se especifica la variable, toma por defecto la variable simbólica principal (la definida por el comando symvar)
collect(‘expr’, ‘[x,y])’)	Agrupa la expresión algebraica polinómica en potencias ordenadas delas variables x e y
collect(‘expr’, ‘f(x)’)	Agrupa la expresión algebraica en potencias ordenadas de una función f(x) contenida en la expresión
[n,m]=numden(‘expr-rat’)	Da numerador y denominador de la expresión racional simplificada

2.2 Ecuaciones Polinómicas y Polinomios

Matlab permite el trabajo ágil con polinomios. El programa habilita varios comandos para el manejo algebraico de las expresiones polinómicas. Estas expresiones pueden tratarse también como expresiones algebraicas en general, pero Matlab particulariza el estudio para las expresiones algebraicas polinómicas incluyendo comandos específicos para el caso. Veamos algunos de estos comandos.

roots(a)	Da las raíces del polinomio cuyos coeficientes son las componentes del vector a
poly(v)	Da el polinomio cuyas raíces son las componentes del vector v
poly(A)	Da al polinomio característico de la matriz A
polyval(p,S)	Evalúa el polinomio p en S
polyvalm(p,S)	Evalúa matricialmente el polinomio p en S
conv(a,b)	Realiza el producto de los polinomios cuyos coeficientes son los elementos de los vectores a y b
[q,r]=deconv(a,b)	Cociente y resto de la división polinómica de a entre b
[n,d,q]=residue(a,b)	Cociente q y expansión del resto en fracciones simples, siendo n y d los respectivos numeradores y denominadores. Cuando el denominador tiene raíces múltiples, la repetición de polos en d indica las potencias de la fracción simple.
[a,b]=residue(n,d,q)	Realiza la operación inversa de la anterior, obteniendo el numerador y denominador a partir de la expansión en fracciones simples.
sym2poly(poli)	Convierte el polinomio simbólico poli en un vector cuyas componentes son sus coeficientes
poly2sym(vector)	Convierte el vector en un polinomio cuyos coeficientes son las componentes del vector
polyder(a)	Da el vector cuyos coeficientes son los del polinomio primera derivada del polinomio a
polyder(a,b)	Da el vector cuyos coeficientes son los del polinomio derivada del producto de a y b
[q,d]=polyder(a,b)	Da la derivada del polinomio cociente de a entre b

3 Interpolación y Aproximación

polyfit(x,y,n)	Da el vector de coeficientes del polinomio en x, p(x) de grado n que mejor ajusta los datos (xi,yi) en el sentido de mínimos cuadrado (p(xi)=yi)
yi=interp1(X,Y.Xi, ‘método’)	Da el vector Yi tal que (Xi, Yi) es el conjunto total de puntos hallados por interpolación entre el conjunto de puntos dados (X,Y). La opción método puede tomar los valores linear, spline o cubic, según que la interpolación sea lineal (opción por defecto), escalonada o cúbica (puntos xi uniformemente separados). Interpolación en una dimensión
zi=interp2(X,Y,Xi,Yi ‘método’)	Da el vector Zi, que determina los puntos de interpolación (Xi, Yi, Zi) entre los puntos dados (X,Y,Z). Se utiliza un método de la distancia inversa para interpolar.
Y= interpft(X,n)	Da el vector Y que contine los valores de la función periódica X muestreada en n puntos igualmente espaciados. El vector original x es transformado al dominio de frecuencias de Fourier utilizando la transformada rápida de Fourier (algoritmo FFT). Ha de cumplirse que n? lengthx

3.1 Polinomios Ortogonales

T(n,x)	Polimomios de chebychev de primera clase
U(n,x)	Polinomios de chebychev de segunda especie
P(n,x)	Polinomios de Legendre
H(n,x)	Polinomios de Hermite
L(n,x)	Polinomios de Laguerre
L(n,a,x)	Polinomios generalizados de Laguerre
P(n,a,b,x)	Polinomios de Jacobi
G(n,m,x)	Polinomios de Gegenbauer

3.1.1 Polinomio de Taylor

taylor('f',n)	Ofrece el desarrollo de McLaurin de f de orden n

El teorema de Taylor permite, bajo ciertas condiciones, aproximar, alrededor de un punto x_0, una función f(x) por medio de un polinomio.

La orden de MATLAB taylor (f, ´ x=a´, n) nos calcula el polinomio de Taylor de la función f, de orden (n-1) y centrado en el punto x=? .

El resto que maneja MATLAB es una ómicron mayúscula (O), expresión acotada cuando se pasa al límite.

x=nnls(A,b)	Resuelve A*x=b en el sentido de mínimos cuadrados, siendo x un vector (x?0)
x=lscov(A,b,v)	Da x (vector) que minimiza (A*x-b)´inv(V)*(A*x-b)

4 Diferenciación e Integración

La derivación e integración son dos operaciones básicas en el estudio y las aplicaciones del Cálculo, además de ser utilizadas con asiduidad en multitud de disciplinas de la ingeniería. Para poder aplicar los operadores derivación e integración a una función es necesario definirla,, `previamente, de una manera simbólica.

4.1 Cálculo de derivadas

Matlab ofrece comandos que permiten el cálculo de derivadas.

diff(‘f’ , ‘x’)e Halla la función derivada de f respecto a x. Si no se especifica la variable de derivación, deriva con respecto a la variable determinada por symvar

diff(‘f’ , ‘x’,n) Halla la función derivada enésima de f con respecto a x

diff(f(x,y,z,...), ‘x’) Define la derivada parcial de f respecto a x

diff(f(x,y,z,...), ‘x’, n) Define la derivada n-ésima de f respecto a x

int(f(x), ‘x’) Calcula la integral indefinida $\int f(x)dx$

int(f(x), ‘x’, ‘a’, ‘b’) Calcula la integral indefinida $\int_a^b f(x)dx$

int(f(x,y), ‘x’, ‘a’, ‘b’), ‘y’, ‘c’, ‘d’))

Calcula la integral definida $\int_a^b \int_c^d f(x,y)dx\, dy$

int(int(int(....(f(x,y,...,z), ‘x’, ‘a’, ‘b’), ‘y’, ‘c’, ‘d’),...), ‘z’, ‘e’, ‘f’)

Calcula $\int_a^b \int_c^d \dots \int_e^f f(x,y,...,z)dx\, dy...dz$

int(f(x), ‘x’) Calcula la integral indefinida $\int f(x)dx$

int(int(f(x,y), ‘x’), ‘y’) Calcula la integral indefinida $\int \int f(x,y)dx\, dy$

int(int(int(f(x,y,z), ‘x’), ‘y’), ‘z’)

Calcula la integral $\int \int \int f(x,y,z)dx\, dy\, dz$

int(int(int(....(f(x,y,...,z), ‘x’), ‘y’), ‘z’)

Calcula la integral $\int \int \dots \int f(x,y,...,z)dx\, dy...dz$

int(int(f(x,y)m ‘x’, ‘a’, ‘b’), ‘y’, ‘c’, ‘d’))

Calcula la integral definida $\int_a^b \int_c^d f(x,y)dx\ dy$

int(int(int(...int(f(x,y,...,z), ‘x, ‘a’, ‘b’), ‘y’, ‘c’, ‘d’),...), ‘z’, ‘e’, ‘f’)

Calcular la integral $\int_a^b \int_c^d \int_e^f f(x,y,\ldots,z)dxdy\ldots dz$

5 EDO Valor Inicial

El número de comandos que implementa Matlab relativos a este tema no es muy elevado, pero sí muy eficiente. De todas formas, es posible seguir con el programa los métodos algebraicos de resolución ya conocidos para cada tipo de ecuación diferencial. También se implementan métodos de resolución aproximados de ecuaciones y sistemas de ecuaciones diferenciales.

El comando básico para resolución diferenciales es dsolve. Este comando computa soluciones simbólicas de ecuaciones diferenciales ordinarias y sistemas. Las ecuaciones son especificadas por expresiones simbólicas conteniendo la letra D para denotar diferenciación, o las letras D2,D3,...,etc, para denotar diferenciación de orden 2,3,..,etc. A Continuación de la letra D se sitúa la variable dependiente (que suele ser y), y cualquier letra no precedida por D es un candidato a variable independiente. Si no se especifica la variable independiente, por defecto es x. Si x se especifica como variable dependiente, la variable independiente es t. O sea, x es la variable independiente por defecto, y en segundo lugar t.

Se puede especificar condiciones iniciales en ecuaciones adicionales, mediante la forma y(a)=b o Dy(a)=b,...,etc. Si las condiciones iniciales no se especifican, las soluciones de las ecuaciones diferenciales contienen constantes de integración C1, C2,..., etc. Los comandos más importantes de Matlab que resuelven ecuaciones diferenciales son los siguientes:

dsolve('ecuación','v')	Resuelve la ecuación diferencial siendo v la variable independiente (si no se especifica 'v', la variable independiente es x). Sólo devuelve soluciones explícitas
dsolve('ecuación', 'condión_inicial',..., 'v')	Resuelve la ecuación diferencial sujeta a la condición inicial específicada

6 Sistemas Lineales Directos

Para recordar el manejo de las variables vectoriales, es conveniente repasar el apartado 1.2.1.

6.1 Operaciones Básicas

a={a1,a2,...,an}, b={b1,b2,..,bn} c=escalar

a+c=[a1+c a2+c,...an+c]	Suma de una escalar y un vector
a*c=[a1*c a2*c,...an*c]	Producto de una escalar por un vector
a+b=[a1+b1 a2+b2 ....an+bn]	Suma de dos vectores
a.* b=[a1*b1 a2*b2 ....an*bn]	Producto de dos vectores
a-/ b=[a1/b1 a2/b2 ....an/bn]	Cociente a la derecha de dos vectores
a.\ b=[a1\b1 a2\b2 ....an\bn]	Cociente a la izquierda de dos vectores
a.^ b=[a1^b1 a2^b2 ....an^bn]	Vector elevado a escalar
c.^ a=[c^a1 c^a2 ---c^an]	Escalar elevado a vector
a.^ b=[a1^b1 a2^b2 ...an^bn]	Vector elevado a vector

A+B	Suma de las matrices A y B
A-B	Diferencias de las matrices A y B (A menos B)
c*M	Producto del escalar c por la matriz M
A*B	Producto de las matrices A y B (A por B)
A^p	Matriz A elevada a la potencia escalar p
p^A	Escalar p elevado a la matriz A

6.2 Operaciones de matrices

expm(A)	eA calculada a través de autovoalres
expm1(A)	eA calculada a través de aproximantes de padé
expm2(A)	eA calculada a través de series de Taylor
expm3(A)	eA calculada a través de la condición de la matriz de autovectores
logm(A)	Logaritmo neperiano de la matriz A
sqrtm(A)	Raíz cuadrada de la matriz cuadrada A
funm(A,'función')	Aplica la función a la matriz cuadrada A
transpose(A) o A'	Matriz transpuesta de A
inv(A)	Matriz inversa de la matriz cuadrada A (A-1)

det(A)	Determinante de la matriz cuadrada A
rank(A)	Rango de la matriz A
trace(A)	Suma de los elementos de la diagonal de A
svd(A)	Da el vector V de valores singulares de A. Los valores singulares de A son las raíces cuadradas de los autovalores de la matriz simétrica A’ A
[U,S,V]=svd(A)	Da la matriz diagonal S de valores singulares de A (ordenados de mayor a menor), y las matrices U y V tales que A= U*S*V’
cond(A)	Da la condición de la matriz A (cociente entre el mayor y el menor valor singular de A)
rcond(A)	Recíproco de la condición de la matriz A
norm(A)	Norma de A (mayor valor singular de la matriz A)
norm(A,1)	1-norma de A (mayor suma de las columnas de A)
norm(A, inf)	Norma infinita de A (mayor suma de la filas de A)
norm(A, ‘fro’)	F-norma de A, definida por sqrt(sum(diag(A’A)))
Z=null(A)	Da una Base ortonormal del núcleo de A (Z’Z=I). El número de columnas de Z es la nulidad de A
Q=orth(A)	Da una base ortonormal para el rango de A (Q’Q=I). Las columnas de Q generan el mismo espacio que las columnas de A, y el número de columnas de Q es el rango de A
subspace(A,B)	Da el ángulo entre los subespacios especificados por las columnas de A y de B. Si a y B son vectores da el ángulo formado por ambos.
rref(A)	Da la matriz reducida escalonada por filas de A. El número de filas no nulas de rref(A) es el rango de la matriz a A

6.3 Resolución de Sistemas

X=linsolve(A,B)	Resuelve A*X =B para una matriz cuadrada A, siendo B y X matrices
X=A\B	Resuelve el sistema A*X=B
X=A/B	Resuelve el sistema X*A=B

6.4 Factorizaciones

vander(C)	Devuelve la matriz de Vandermonde A tal que su j-ésima columna es A(:,j)=C^(n-j)
[L,U]=LU(A)	Descompone la matriz A en el producto A =L*U (descomposición LU de A), siendo U una matriz triangular superior y L una matriz pseudotriangular inferior (triangularizable mediante permutación)
[L,U,P]=LU(A)	Devuelve una matriz triangular inferior L, una matriz triangular superior U, y una matriz de permutación P tales que P*A =L*U
R=chol(A)	Devuelve la matriz triangular superior R tal que R´*R=A (descomposición de Cholesky de A), en caso de que A sea definida positiva. Si A no es definida positiva devuelve un error.
[Q,R]=qr(A)	Devuelve la matriz triangular superior R de la misma dimensión que A, y la matriz ortogonal Q tales que A=Q*R(descomposición QR de A). Esta descomposición puede aplicarse a matrices no cuadradas.
[Q,R.E]=qr(A)	Devuelve la matriz triangular superior R de la misma dimensión que A, la matriz permutación E y la matriz ortogonal Q tales que A*E=Q*R
X=pinv(A)	Devuelve la matriz X (pseudoinversa de A), de la misma dimensión que A´ tal que A*X*A=A y X*A*X=X, siendo A*X y X*A matrices hermitianas.

6.5 Operaciones con matrices simbólicas

sysub(A,B)	Diferencia de las matrices A y B (A menos B)
sympow(A,p)	Matriz A elevada a la potencia escalar p
transpose(A)	Matriz transpuesta de A (A’)
inverse(A)	Matriz inversa de la matriz cuadrada A (A-1)
determ(A)	Determinante de la matriz cuadrada A

[U,S,V]=singvals(A) o [U,S,V]=svdvpa(A)

Devuelve las matrices ortogonales u y V y la matriz diagonal S con los valores singulares de A en la diagonal, tales que A =USV’

symop(A, ‘operación1’, B, ‘operación2’,C,...

Realiza las operaciones indicadas entre las matrices simbólicas dadas y en el orden especificado. Este comando permite mezclar todo

6.6 Autovalores y Autovectores

Matlab habilita comandos que permiten trabajar con autovalores y autovectores de una matriz cuadrada. Para matrices numéricas, tenemos los siguientes.

eig(A) Calcula los autovalores de la matriz cuadrada A

[V,D]=eig(A) Calcula la matriz diagonal D de autovalores de A y una matriz V cuyas columnas son los autovectores

eig(A,B) Da un vector que contiene los autovalores generalizados de las matrices cuadradas A y B. Los autovalores generalizados de A y B son las raíces del polinomio en ? det(?*C-A)

[V,D]=eig(A,B) Calcula la matriz diagonal D de autovalres generalizados de A y B, y una matriz V cuyas colulmnas son los autovectores correspondientes, cumpliéndose que A*V=B*V*D

[AA, BB, Q, Z, V]=qz(A,B)

Calcula las matrices triangulares superiores AA y BB y las matrices Q y Z tales que Q*A*Z=AA y Q*B*Z=BB, y da la matriz V de autovectores generalizados de A y B.Los autovalres generalizados son los elementos de la diagonal de AA y BB, de tal modo que se tiene la igualdad: A*V* diag(BB)=B*V*diag(AA)

[T,B]=balance(A) Encuentra una matriz de transformación T tal que B=T\A*T tien autovalores aproximados a los de A . La matriz B se llama matriz balanceada de la matriz A

balance (A) Calcula la matriz B balanceada de la matriz A. Su uso

	esencial es aproximar los autovalores de A cuando son difíciles de calcular. se tiene que eig(A)=eig(balance(A))
[V,D]=cdf2rdf(V,D)	Transforma la salida compleja [V,D] del comando eig a forma real. cada autovalor complejo en la diagonal de D origina una submatriz 2x2 en la forma real de la matriz D
[U,T]=schur(A)	Da una matriz T y una matriz unitaria U tales que A= U*T*U´ and U´*U=eye(U). Si A es compleja, T es una matriz triangular superior con los autovalores de A en la diagonal. Si A es real, la matriz T tiene los autovalores de A en la diagonal, y los correspondientes autovalores complejos se corresponden con los bloques diagonales 2x2 de la matriz T
schur(A)	Devuelve la matriz T solamente
[U,T]=rsfwcsf(U,T)	Convierte a real la salida [U,T] del comando schur
[P,H]=hess(A)	Devuelve la mamtriz unitaria P y la matriz de Hessenberg H tales que A = P*H*P´ y P´*P=eye(size(P))
hess(A)	Devuelve la matriz de Hessenberg H
poly(A)	Devuelve el polinomio característico de la matriz A
poly(V)	Devuelve un vector cuyas componentes son los coeficientes del polinomio cuyas raíces son los elementos del vector V

[V,E]=eigensys(A)	Devuelve el vector E, conteniendo los autovalores de A, y la matriz V, que contiene sus autovectores
jordan(A)	Devuelve la forma canónica de Jordan, J, de la matriz A (numérica o simbólica). J tiene los autovalores de A en la diagonal
[V,J]=jordan(A)	Devuelve la forma canónica de Jordan,J, de la matriz A, y la matriz de paso Vcuyas columnas son los autovectores de A, cumpliéndose que V1*A*V=J

6.7 Matrices Dispersas

S=space(i,j,s,m,n,nzmax) i=vector, j=vector, s=vector

con	Crea una matriz dispersa S de dimensión mxn espacio para

	nxmax elementos no nulos. el vector i contiene las componentes i-ésima de los elementos no nulos. El vector i contiene las componentes i-ésimas de los elementos no nulos y el vector j contiene sus correspondientes componentes j-ésimas
S=space(i,j,s,m,n)	Crea la matriz dispersa S usando nzmax=legnth(s)
S=space(kmj,s)	Crea la matriz dispersa S usando con m=max(i) y n=max(j)
S=space(A)	Convierte la matriz completa A a la dispersa S
A=full(S)	Convierte la matriz dispersa S en la completa A
S=spconvert(D)	Convierte un fichero ASCII exterior leído con nombre D a la matriz dispersa S
(i,j)=find(A)	Devuelve los índices de filas y columnas no cero de la matriz A
B=spdiags(A,d)	Construye la matriz dispersa deducida de a cuya diagonal son de los elementos del vector d
S=speye(n)	Construye la matriz dispersa identidad cuadrada de orden n
R=sprandn(S)	Genera una matriz dispersa aleatoria de valores normales (0,1) con la misma estructura de la matriz dispersa S
R=sprandsym(S)	Genera una matriz simétrica aleatoria de valores normales(0,1) cuyo triángulo inferior y diagonal tienen la misma estructura que en S
r=sprank(S)	Da el rango estructural de la matriz dispersa S
n=nnz(S)	Da el número de elementos no nulos de la matriz dispersa S
k=nzmax(S)	Da la cantidad de almacenamiento ocupada por los elementos no nulos de la matriz dispersa S. Se tiene que nzmax(S)=prod(size(S))
s=spalloc(m,n,nzmax)	Crea espacio en memoria para una matriz dispersa de dimensión mxn
R=spones(S)	Reemplaza los elementos no nulos de la matriz dispersa s con unos.
n=condest(S)	Devuelve la 1-norma de la matriz dispersa S
m=normest(S)	Devuelve la 2-norma de la matriz dispersa S
issparse(A)	devuelve 1 si la matriz A es dispersa y 0 en otro caso

6.8 Matrices Especiales

H=hadamard(n)	Matriz con valores 1 o-1 tal que H´*H=n*eye(n)
hankel(V)	Matriz cuya primera columna es el vector V y cuyos elementos son cero por debajo de la primera antidiagonal. La matriz hankel(C,R) tiene como primera columna el vector c y como última fila el vector R
hilb(n)	Matriz de Hilbert de orden n tal que Aij=1/(i+j-1)
invhilb(n)	Inversa de la matriz de Hilbert de orden n
magic(n)	Matriz mágica de orden n. Sus elementos son los enteros desde 1 hasta n2. con iguales sumas de filas y columnas
pascal(n)	Matriz de Pascal de orden n(simétrica, definida positiva y basada en el triángulo de Pascal)
rosser	Matriz 8x8 con un autovalor doble, otro nulo, tres casi iguales, otro muy pequeño y dos opuestos
toeplitz(C,R9	Matriz de Toeplitz (no simétrica con el vector c de primera columna y el vector R como primera fila)
vander(C)	Matriz de Vandermonde cuya penúltima columna es el vector C. Además, A(:,j)=C^(n-j)
wilkinson(n)	Matriz de Wilkinson (simétrica tridiagonal con pares de autovalores cercanos pero no iguales)
compan(P)	Matriz del polinomio de coeficientes P

6.9 Resolución de ecuaciones y sistema

Matlab ofrece determinados comandos que permiten resolver ecuaciones y sistemas. Entre ellos tenemos los siguientes:

solve(‘ecuación’, ‘x’) Resuelve la ecución en la variable x

solve (‘ex1,ex2,...,ecn’, ‘x1,x2,...,xn’)

Resuelve n ecuaciones simultáneas ec1,...,ecn en las variables x1,...,xn (sistema de ecuaciones)

solve(‘ecuación’, ‘x’)	Resuelve la ecución en la variable x
solve (‘ex1,ex2,...,ecn’, ‘x1,x2,...,xn’)	Resuelve n ecuaciones simultáneas ec1,...,ecn en las variables x1,...,xn (sistema de ecuaciones)
X=linsolve(A,B)	Resuelve A*X =B para una matriz cuadrada A, siendo B y X matrices
x=nnls(A,b)	Resuelve A*x=b en el sentido de mínimos cuadrados, siendo x un vector (x?0)
x=lscov(A,b,v)	Da x (vector) que minimiza (A*x-b)´inv(V)*(A*x-b)
roots(V)	Da las raíces del polinomio cuyos coeficientes son las componentes del vector V.
X=A\B	Resuelve el sistema A*X=B
X=A/B	Resuelve el sistema X*A=B

6.10 Operaciones Matriciales

diag(v)	Crea la matriz identidad de orden n
diag(A)	Extraer la diagonal de la matriz A como vector columna
eye(n)	Crea la matriz identidad de orden n-
eye(m,n)	Crea la matriz de orden mxn con unos en la diagonal principal y ceros en el resto.
zeros(m,n)	Crea la matriz nula de orden mxn
ones(m,n)	Crea la matriz de orden mxn con todos sus elementos 1
rand(m,n)	Crea una matriz aleatoria uniforme de orden mxn
randn(m,n)	Crea una matriz aleatoria normal de orden mxn
flipud(A)	Devuelve la matriz cuyas filas están colocadas en orden inverso (de arriba abajo) a las filas de A
fliplr(A)	Devuelve la matriz cuyas columnas están colocadas en orden inverso /de izquierda a derecha) a las de A
rot90(A)	Rota 90 grados la matriz A
reshape(A,m,n)	Devuelve la matriz de orden mxn extraída de la matriz A tomando elementos consecutivos de A por columnas
size(A)	Devuelve el orden (tamaño) de la matriz A
find(condA)	Devuelve los elementos de A que cumplen la condición

length(v)	Devuelve la longitud del vector v
tril(A)	Devuelve la parte triangular inferior de la matriz A
triu(A)	Devuelve la parte triangular superior de la matriz A

6.11 Operaciones Matriciales

A+B,A-B,A*B	Suma, resta y producto de matrices
A\B	Si A es cuadrada A\B=inv(A)*B. Si A no es cuadrada A\B es la solución en el sentido de mínimos cuadrados del sistema AX=B
B/A	Coincide con (A’ \ B’)’
An	Coincide con A*A*A*.....*A n veces (n escalar)
pA	Realiza el cálculo sólo si p es un escalar

6.12 Funciones Matriciales

max(V)	Mayor componente (para complejos se calcula max(abs(V)))
min(V)	Mayor componente (para complejos se calcula min(abs(V)))
mean(V)	Media de los componentes de V
median(V)	Mediana de la s componentes de V
std(V)	Desviación típica de las componentes de V
sort(V)	Ordena de forma ascendente las componentes de V. Para complejos hace la ordenación según los valores absolutos
sum(V)	Suma los componentes de V
prod(V)	Multiplica los elementos de V, con lo que n!=prod(1:n)
cumsum(V)	Da el vector de suma acumuladas de V
cumprod(V)	Da al vector de productos acumulados de V
diff(V)	Da el vector de primeras diferencias de V (Vt-Vt-1)
gradient(V)	Aproxima el gradiente de V
del2(V)	Laplaciano de V (discreto de 5 puntos)
fft(V)	Transformada discreta de Fourier V
fft2(V)	Transformada discreta bidimensional de Fourier de V
ifft(V)	Inversa de la transformada discreta de Fourier de V
ifft2(V)	Inversa de la transformada 2-D discreta de Fourier de V

6.13 Operaciones Lógicas sobre Matrices

exit(A)	Chequea si la variable o función A existe (devuelve 0 si A no existe, y un número entre 1 y 5, según el tipo, si existe)
any(V)	Devuelve 0 si todos los elementos del vector V son nulos, y devuelve 1 si algún elemento de V es no nulo
all(V)	Devuelve 1 si todos los elementos del vector V son no nulos , y devuelve 1 si algún elemento de V es no nulo
find(V)	Devuelve los lugares (o índices) que ocupan los elemento no nulos del vectro V
isnan(V)	Devuelve 1 para los elementos de V que son indeterminados, y devuelve 0 para los que no lo son
isinf(V)	Devuelve 1 para los elementos de V que son infinitos, y devuelve 0 para los que no lo son
finite(V)	Devuelve 1 para los elementos de V que son finitos, y devuelve 0 para los que no lo son
isempty(A)	Devuelve 1 si A es una matriz vacía, y devuelve 0 en otro caso (una matriz vacía es la que tiene una de sus dimensiones 0)
issparse(A)	Devuelve 1 si A es una matriz por cajas, y devuelve 0 en otro caso
isreal(V)	Devuelve 1 si todos los elementos de V son reales, y 0 en otro caso
ishold	Devuelve 1 si retienen las propiedades del gráfico actual para el siguiente y sólo se añaden las nuevas, y 0 en caso contrario.
isieee	Devuelve 1 para computadores IEEE
isstr(S)	Devuelve 1 si S es una cadena, y 0 en caso contrario
isglobal(A)	Devuelve 1 si A es una variable global, y 0 en otro caso
isletter(S)	Devuelve 1 si S es una letra de alfabeto, y 0 en otro caso

6.14 Funciones Especiales

besselj(V,X)	Función de Bessel de primer clase (V y X vectores)
bssely(V,X)	Función de Bessel de segunda clase (V y X vectores)
besseli(V,X)	Función de Bessel modificada de primer clase (V y X vectores)

besselk(V,X)	Función de Bessel modificada de segunda clase (V y X vectores)
beta(x,y)	Función Beta
betainc(X,A,B)	Función Beta incompleta (X vector y A y B matrices)
betaln(x,y)	Logaritmo de la función Beta
ellipj(A,B)	Función elíptica de Jacobi (A y B matrices)
ellipke(A)	Integral elíptica completa (A matriz)
erf(x)	Función error
erfc(x)	Complementario de la función de error
erfcx(x)	Complementario a escala de la función de errro
erfinv(x)	Inversa de la función de eror
expint(X)	Función integral exponencial (X vector)
gamma(X)	Función gamma (X vector)
gammainc(X,Y)	Función gamma incompleta (X e Y vectores)
gammaln(X)	Logaritmo de la función Gamma (X vector)
gcd(x,y)	Máximo común divisor (x e y números enteros)
lcm(x,y)	Mínimo común múltiplo (x e y números enteros)
legendre(n,X	Función asociada de Legendre (n entero y X vector)
log2(x)	Logaritmo en base 2 de x
pow2(x)	Potencia de base 2 de x

6.15 Operaciones con matrices

La matriz $A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & & a_{mn} \end{pmatrix}$ i=1,...,m j=1,...n, se introduce en Matlab de las siguientes formas:

Numérica

$A = [a_{11} a_{12} a_{13} \dots a_{1n}; a_{21} a_{22} a_{23} \dots a_{2n}; \dots; a_{m1} a_{m2} a_{m3} \dots a_{mn}]$

$A = [a_{11}, a_{12}, a_{13}, \dots, a_{1n}; a_{21}, a_{22}, a_{23}, \dots, a_{2n}; \dots; a_{m1}, a_{m2}, a_{m3}, \dots a_{mn}]$

Simbólica

$$A = [a_{11} a_{12} a_{13} \dots a_{1n}; a_{21} a_{22} a_{23} \dots a_{2n}; \dots ; a_{m1} a_{m2} a_{m3} \dots a_{mn}]$$

$$A = sym([a_{11}, a_{12}, a_{13}, \dots , a_{1n}; a_{21}, a_{22}, a_{23}, \dots , a_{2n}; \dots ; a_{m1}, a_{m2}, a_{m3}, \dots a_{mn}])$$

Operación con matrices numéricas

A continuación se presentan comandos de Matlab que permiten las operaciones con matrices numéricas:

A+B	suma las matrices A y B.
A-B	diferencia de las matrices A y B.
C*M	producto del escalar c por la matriz M-
A*B	producto de las matrices A y B.
A^P	matriz A elevada a la potencia escalar p.
transpose (A) o A’	matriz transpuesta de A.
inv(A)	matriz inversa de la matriz cuadrada A
det(A)	determinante de la matriz cuadrada A
rank(A)	rango de la matriz A.
trace(A)	traza A.

Operaciones con matrices simbólicas.

A continuación se presentan comandos de Matlab que permiten las operaciones con matrices simbólicas.

symadd(A,B)	suma las matrices A y B.
symsub(A,B)	diferencia de las matrices A y B.
symmul(A,B)	producto de las matrices A y B.
sympow(A,B)	matriz A elevada a la potencia escalar p.
transpose(A)	matriz transpuesta de A.
inverse(A)	matriz inversa de la matriz cuadrada A
determ(A)	determinante de la matriz cuadrada A

sympo(A,’operación1’,B,’operación2’,C,...) realiza las operaciones indicadas entre las matrices simbólicas dadas y en el orden especificado. Este comando permite mezclar todo tipo de operaciones entre matrices simbólicas.

Ejercicio 6.1 Dada la matriz $A = \begin{pmatrix} \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{pmatrix}$, halla la transpuesta, su inversa, su determinante su traza, A^3

Ejercicio 6.2 Dadas las matrices:

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} i & 1-i & 2 ? i \\ 0 & 1 & 3 ? i \\ 0 & 0 & ? i \end{pmatrix} \quad C = \begin{pmatrix} 1 & 1 & 1 \\ 0 & \sqrt{2i} & ? \sqrt{2i} \\ 1 & ? 1 & ? 1 \end{pmatrix}$$

calcula AB-BA, $A^2+B^2+C^2$,ABC,3A+2B y halla la inversa, la traza, el determinante y la matriz traspuesta de A,B y C.

Ejercicio 6.3 Dada la matriz

$$A = \begin{pmatrix} a & b & c \\ 3c & a ? 3c & b \\ 3b & ? 3b ? 3c & a ? 3c \end{pmatrix}$$

Calcula traspuesta de A, A^{-1}, determinante de A, traza de A, a^2 y la matriz adjunta de A.

Ejercicio 6.4 Dadas las matrices A y B siguientes:

$$A = \begin{pmatrix} \cos(a) & \operatorname{sen}(a) \\ \operatorname{sen}(a) & \cos(a) \end{pmatrix} \text{ y } \quad B = \begin{pmatrix} \operatorname{sen}(a) & \cos(a) \\ \cos(a) & \operatorname{sen}(a) \end{pmatrix}$$

Calcula $M1=A^2+B^2$ y $M2=A^2+B^2$

6.16 Espacios vectoriales

En esta práctica vamos a manejar los comandos matriciales ya conocidos. así como otro comando específico que se relacionan a continuación.

colspace(A): devuelve una base para las columnas de A.

Ejercicio 6.5a)En el espacio vectorial (R^3,+R), estudiar si el subconjunto S={u_1,u_2,u_3}es ligeramente independiente, siendo u_1=(2,3,-1), u_2=(0,0,1),u_3=(2,1,0).

b)En el espacio vectorial (R^4,+,R) estudiar si los subconjuntos s={ u_1,u_2,$u_3$$u_4$} y S´{$v_1$,$v_2$,$v_3$} son linealmente independientes, siendo u_1=(1,3,-3,4), u_2=3,-1,2,1), u_3=(1,-5,8,-7), u_4=(2,3,1,-1), v_1(1,2,2,1),v_2=(3,4,4,3), v_3=(1,0,0,1).

Ejercicio 6.6 Dados los vectores v_1=(2,3,4,-1,1), v_2=(3,4,7,-2,-1),v_3=(1,3,-1,1,8),v_4=(0.5,5,-1,4). Obtén la dimensión del subespacio engendrado por ellos y una base de dicho subespacio.

Ejercicio 6.7 Dado el subconjunto de vectores del espacio vectorial (R^3,+,R) B={(2,3,-1), (0,0,1),(2,1,0)},estudiar si forman una base y en caso positivo, obtener las componentes del vector x=(3,5,1)en dicha base.

Ejercicio 6.8 Consideremos las bases B_1 y B_2 del espacio vectorial real tridimensional B_1={(1,0,0),(-1,1,0),(0,1,-1)} y B_2={(1,0,-1),(2,1,0)(-1,1,**1}**

Calcular la matriz del cambio de base de B_1 a B_2- M[B_1,B_2]- y calcular las componentes de 1 vector v en B_2, sabiendo que en B_1 son (2,1,3).

Ejercicio 6.9 Sea V un espacio vectorial de dimensión 5, **B=**{ $\mathbf{u_1,u_2,u_3 ,u_4,u_5}$} una base de V. Sea F el subespacio engendrado por los vectores $\mathbf{a_1=u_1-u_2+3u_3+2u_4+5u_5,a_2= u_1+u_2+u_3 +2u_4+3u_5,a_3= u_1+u_3-u_4 +2u_5}$ y G el subespacio engrendrado por los vectores $\mathbf{b_1= 13u_1+2u_2+3u_3 +8u_4+8u_5}$,$\mathbf{b_2= 17u_1+3u_2+3u3+10u_4+9u_5}$**.** Halla una base para F+G y la dimensión de F? G.

6.17 Aplicaciones lineales. Sistemas de ecuaciones lineales.

En esta práctica vamos a manejar los comandos matriciales ya conocidos, así como otros comandos específicos que se relacionan a continuación:

cospace(A):devuelve una base para las columnas de A.

nullspace(A): devuelve una base para el núcleo de la aplicación lineal cuya matriz es A.

N=null(A): devuelve una base para el núcleo de la aplicación lineal cuya matriz es A.

Solver (‘ecuación’,’x’): resuelve la ecuación en la variable x

Solver (‘ec1’,’ec2’,...’ecm’,x1,x2,...xn’): resuelve el sistema de ecuaciones lineales.

X=linsolver(A,B): resuelve AX=B para una matriz cuadrada A.

Ejercicio 6.10 Dada la aplicación lineal f de R^5 en R^3, cuya matriz asociada respecto las bases canónicas es

$$A \,?\, \begin{pmatrix} 0 & ?3 & ?1 & ?3 & ?1 \\ ?3 & 3 & ?3 & ?3 & 1 \\ 2 & 2 & ?1 & 1 & 2 \end{pmatrix}$$

1. Encontrar una base para el núcleo de f y otra para Im(f).
2. Hallar la imagen de los vectores (4,2,0,0,-6) y (1,2,-1,-2,3) mediante la aplicación f.
3. ¿El vector (3,2,4)? Im(f)?

Ejercicio 6.11: Sea f y g las aplicaciones lineales definidas de (R^2,+,R) y de (R^3.+, R) en (R^4,+,R) respectivamente, tales que

f(1,1)=(5,2,3) g(-2,4,2)(1,1,1,1)

f(2,3)=(2,0,4) g(1,0,-1)=(2,-1,3,4)

g(-1,2,0)=(0,1,0,1)

Hallar:

1. La matriz asociada a la aplicación g o f respecto las bases canónicas.
2. La dimensión del núcleo y el rango de la aplicación compuesta.
3. Una base para el núcleo y otra para el subespacio Im(f).
4. Las ecuaciones de g o f.

Ejercicio 6.12 Estudiar y resolver los siguientes sistemas lineales:

a) $$\begin{cases} 2x_1+x_2+x_3+x_4=1 \\ x_1+2x_2+x_3+x_4=1 \\ x_1+x_2+2x_3+x_4=1 \\ x_1+x_2+x_3+2x_4=1 \end{cases}$$

b) $$\begin{cases} x_1-x_2+x_3=1 \\ 4x_1+5x_2-5x_3=4 \\ 2x_1+x_2+x_3=2 \\ x_1+2x_2+-2x_3=1 \end{cases}$$

c) $$\begin{cases} x+2y+3z=6 \\ x+3y+8z= 19 \\ 2x+3y+z=-1 \\ 5x+6y+4z=5 \end{cases}$$

d) $$\begin{cases} x+2y+3z+t=6 \\ x+3y+8z+t=19 \end{cases}$$

e) $$\begin{cases} x+2y-z=10 \\ 2x+4y-2z=5 \\ x+y+z=6 \end{cases}$$

f) $$\begin{cases} x+2y-z=0 \\ 2x-y+z=0 \\ 3x+y=0 \end{cases}$$

g) $$\begin{cases} 3x+y+z-t=0 \\ 2x+y-z+t=0 \\ x+2y+4z+2t=0 \\ 2x+y+2z-t=0 \end{cases}$$

6.18 Operadores lineales. Funciones bilineales y formas cuadráticas.

En esta práctica vamos a manejar los comandos matriciales ya conocidos, así como otros comandos específicos que se relacionan a continuación.

Para matrices numéricas;

eig(A): calcula los autovalores de la matriz cuadrada A.

[**P,D**]=**eig(A)**: calcula la matriz diagonal D de autovalores de A y una matriz P cuyas columnas son los autovectores correspondientes, cumpliendose que P^{-1}*A*P=d.

Poly(A): devuelve el polinomio característico de la matriz A.

Poly(V): devuelve un vector cuyas componentes son los coeficientes del polinomio cuyas raíces son los elementos del vector V.

Para matrices simbólicas:

eigensys(A): devuelve los autovalores de la matriz A.

[P,E]=**eigensys(A):** devuelve el vector E de autovalores de A y una matriz P cuyas columnas son los autovectores correspondientes.

Charpoly (A,'x'): devuelve el polinomio característico de A en función de x, cuyo valor es det (xI-a).

jordan (A): devuelve la forma canónica de Jordan, J, de la matriz A (numérica o simbólica).

[**P,J**]=**jordan(A):**devuelve la forma canónica de Jordan,J, de la matriz A, y la matriz de paso P, tal que P^{-1}*A*P=J

[**L,U**]=**lu(A):**descompone la matriz A en el producto A=L*U (descomponiendo LU de A), siendo U una matriz triangular superior y L una matriz pseudo-triangular inferior(triangularizable mediante permutación.

R=chol(A):devuelve la matriz triangular superior R tal que R'*R=A (descomposición de Cholesky de A), en caso de que A sea definida positiva. Si A no es definida positiva devuelve un error.

Ejercicio 6.13 Determinar los valores y vectores propios de las siguientes matrices, así como su polinomio característico.

$$A = \begin{pmatrix} 2 & 0 & 4 \\ 3 & -4 & 12 \\ 1 & 2 & -5 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \qquad C = \begin{pmatrix} -4 & 0 & -2 \\ 0 & 1 & 0 \\ 5 & 1 & 3 \end{pmatrix}$$

Ejercicio 6.14 Calcule la forma canónica de Jordan y la correspondiente matriz de paso de las siguientes matrices:

$$A\ ?\ \begin{pmatrix} ?1 & ?4 & 0 & ?1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & ?1 & 0 \\ 4 & 7 & ?12 & 3 \end{pmatrix} \qquad B\ ?\ \begin{pmatrix} 1 & 1 & ?1 & 1 & ?1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Ejercicio 6.15 Estudia si existe una matriz diagonal semejante a cada una de las siguientes matrices:

$$A\ ?\ \begin{pmatrix} 0 & ?r & q \\ r & 0 & ?p \\ ?q & p & 0 \end{pmatrix} \quad B\ ?\ \begin{pmatrix} 0 & 1 & ?\,\mathrm{sen}\,? \\ ?1 & 0 & \cos ? \\ ?\,\mathrm{sen}\,? & \cos ? & 0 \end{pmatrix} \quad C\ ?\ \begin{pmatrix} \cos ? & ?\,\mathrm{sen}\,? \\ \mathrm{sen}\,? & \cos ? \end{pmatrix}$$

Ejercicio 6.16 Consideremos la matriz cuadrada de orden 4. Realizar las descomposiciones LU y de Cholesky.

Ejercicio 6.17 Dadas las formas cuadráticas:

$$f(x,y,z)=x^2-2xy+y^2+6xz.3yz+4z^2$$

$$g(x,y,z)=x^2+2y^2+4yz*2z^2$$

Hallar las matrices asociadas y clasificarlas. Hallar sus ecuaciones reducidas, sus rangos y signaturas.

7 Sistemas Lineales Iterativos

8 Aproximación

9 Autovalores

10 Sistemas No Lineales

11 EDO Contorno

12 EDP Contorno

13 Optimización