FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

CURSOS ABIERTOS

DIPLOMADO DE MATEMÁTICAS

TEMA

CA 485

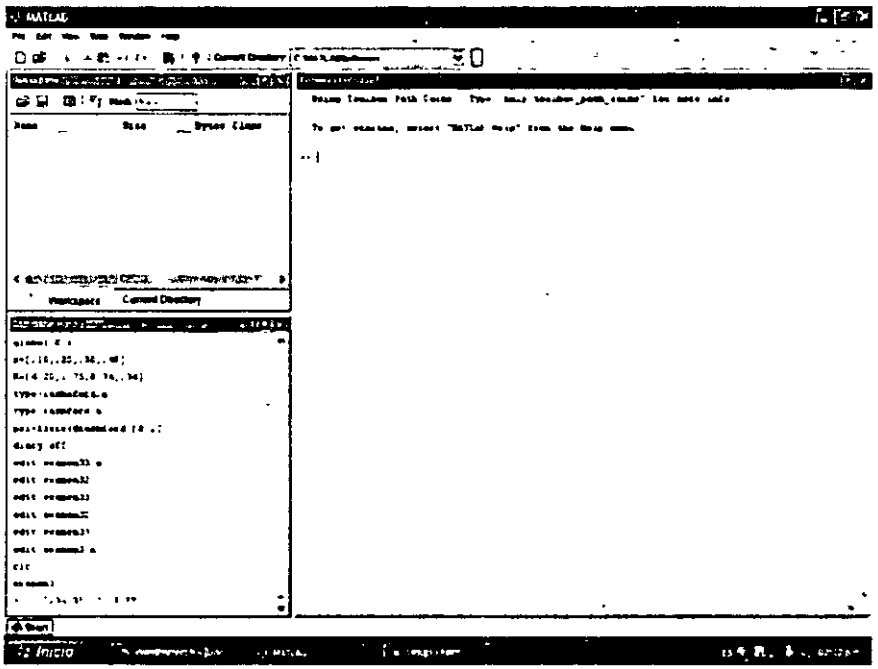
APLICACIONES DE MATEMÁTICAS CON SOFTWARE, MAPLE Y MATHCAD.

INSTRUCTOR: MAT. ARTURO LÓPEZ GARCÍA
DEL 10 DE DICIEMBRE 2005 AL 14 DE ENERO 2006
PALACIO DE MINERÍA

1.- Ambiente de trabajo del MATLAB

Para iniciar el ambiente de trabajo. simplemente ejecute con un doble click el icono

colocado en el escritorio de Windows. Como consecuencia de lo anterior entonces aparece el escritorio de MATLAB. La disposición por omisión de dicho escritorio es

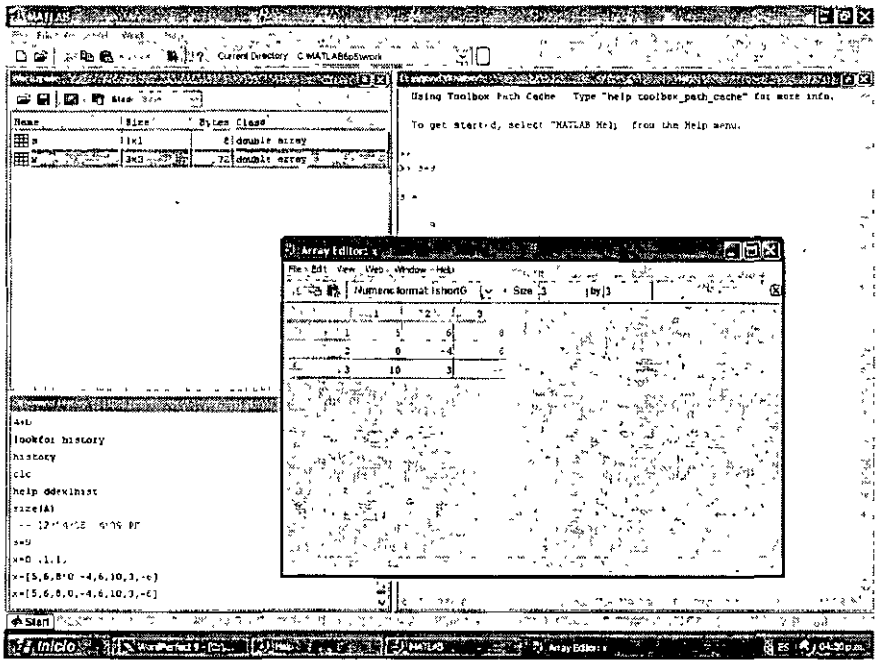


Si aparece otra organización distinta. puede regresar al escritorio por omisión seleccionando `View` en el menú principal. después `Desktop Layout` y luego `Default`.
El escritorio por omisión muestra tres ventanas: `Command Window` ubicada a la derecha. `Workspace` y `Command History` colocadas a la izquierda. en la parte superior e inferior. respectivamente.
En cualquier momento el usuario puede seleccionar cuales de las tres ventanas serán mostradas en el escritorio. basta con indicarlas en la lista mostrada después de ejecutar `View` en el menú principal.
Dentro de la `Command Window` se ejecutan las órdenes de MATLAB y aparecen los resultados correspondientes. El indicador de MATLAB que aparece en esta ventana avisando que

MATLAB está listo para procesar otra orden es >>.
Cada vez que es ejecutada una orden en la Command Window , esta es guardada en la *historia de órdenes*. El contenido de la historia de órdenes es mostrado en la ventana Command History.
Dentro de la Command Window , es posible recuperar el contenido de la historia de órdenes: con las teclas ↑ y ↓ se retrocede o se avanza en ella, respectivamente, escribiendo en la ventana Command Window una copia de la orden seleccionada. En ese momento, el usuario puede modificar y/o ejecutar la orden. Otra forma de recuperar órdenes es seleccionarlas directamente de la Command History : una vez seleccionada la orden puede ser copiada con Ctrl+C y luego pegada en la ventana de órdenes con Ctrl+V. Si se desea ejecutar directamente la orden, basta entonces con presionar Enter después de seleccionarla dentro de la ventana Command History.
El control del cursor en la ventana de órdenes es a través de combinaciones de teclas como las siguientes:

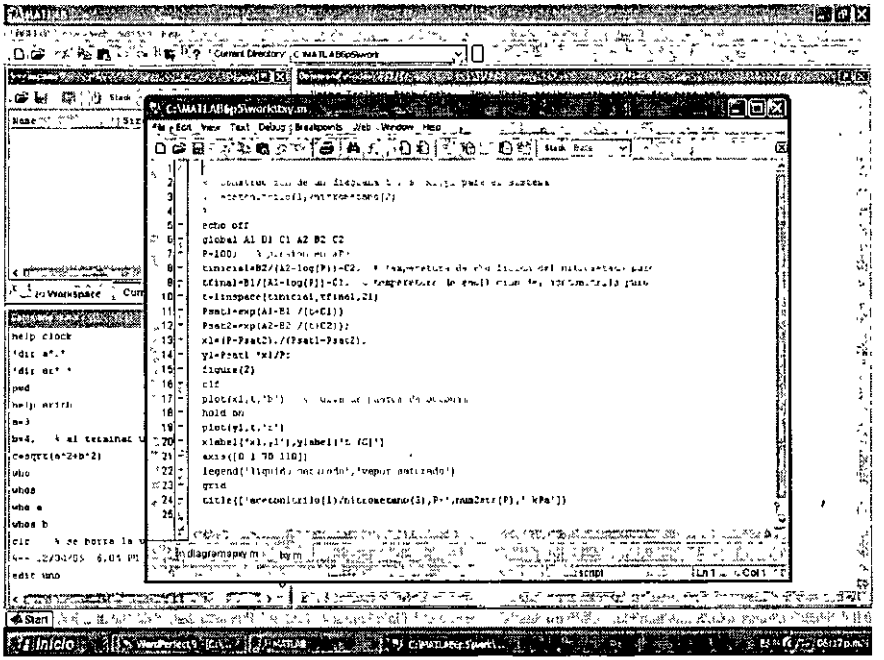
Combinación de teclas	Función
Ctrl+→	Movimiento del cursor hacia el inicio de la siguiente palabra
Ctrl+←	Movimiento del cursor hacia el inicio de la palabra anterior
Esc	Borrado de la línea completa
Shift+Inicio	Selección desde el cursor hasta el inicio de la línea
Shift+Fin	Selección desde el cursor hasta el final de la línea

Por omisión, en la ventana Workspace aparece información sobe las variables que están definidas en el *área de trabajo* (*workpace*). Dicha información consiste en el nombre, el tamaño, el número de bytes que ocupa y el tipo. Una forma de ver el contenido de una variable incluida en el área de trabajo es ejecutar un doble click en el nombre de la misma dentro de la ventana Workspace. Lo anterior arranca una aplicación llamada *Editor de Arreglos* (*Array Editor*) la cual permite revisar y/o modificar el contenido de una variable. Es importante recalcar que MATLAB considera que todas las variables son arreglos; esa es la razón por la cual las variables simples aparecen en la ventana Workspace como arreglos de una fila por una columna.



El contenido de las tres ventana puede ser borrado mediante las funciones `Clear Command Window`, `Clear Command History` y `Clear Workspace`. El acceso a estas es a través de `Edit` en el menú principal.

Dentro del entorno de trabajo de MATLAB es factible crear o modificar cualquier archivo de texto: en el menú principal seleccione `File` y luego `New` u `Open` según sea el tipo de operación por realizar. Surge una ventana titulada con el nombre del archivo y entonces el usuario podrá editar el contenido del mismo.


C:\MATLAB6p5\work\key.m
echo off
global A1 B1 C1 A2 B2 C2
P=100;
tinicial=B2/(A2-log(P))-C2;
tfinal=B1/(A1-log(P))-C1;
t=linspace(tinicial,tfinal,21)
Psat1=exp(A1-B1 /(t+C1))
Psat2=exp(A2-B2 /(t+C2));
x1=(P-Psat2)./(Psat1-Psat2);
y1=Psat1 *x1/P;
figure(2)
clf
plot(x1,t,'b')
hold on
plot(y1,t,'r')
xlabel('x1,y1'),ylabel('t (C)')
axis([0 1 70 110])
legend('liquido saturado','vapor saturado')
grid
title(['azeotropico(1)/nitrometano(3),P=',num2str(P),' kPa'])
diagramapxy.m
help clock
dir a*.*
dir ac* *
pwd
help wcirb
a=3
bw4, % al terminar
cos(pi*(a^2+b^2)
who
whos
who a
whos b
clc % se borra la
edit uno
Start
Inicio

2.- Operaciones básicas con MATLAB

2.1.- Órdenes básicas

MATLAB distingue entre las letras mayúsculas y las minúsculas. Por ejemplo, version es diferente a VERSION o a Version.
Junto al MATLAB básico es posible instalar las llamadas *cajas de herramientas* (*toolboxes*) que son paquetes adicionales diseñados para resolver problemas en un campo específico.

ver	Indica la versión del MATLAB instalado
version	Además de la información proporcionada por ver, da los nombres de las cajas de herramientas instaladas.
help orden	Despliega la información de ayuda disponible para una orden
lookfor texto	Genera una lista con los nombres de las órdenes de MATLAB cuyos archivos de ayuda contiene la cadena de caracteres texto
date	Proporciona la fecha
clock	Muestra la hora y la fecha
!orden del sistema operativo	Ejecuta una orden del sistema operativo
pwd	Imprime la ruta del directorio actual de trabajo
diary archivo	Inicia el registro de las órdenes ejecutadas y de sus resultados en un archivo de texto
diary off	Da por terminado el registro iniciado con diary
clc	Borra el contenido de la ventana de órdenes
quit	Finaliza la sesión de MATLAB

2.2.- Operaciones aritméticas y variables

La orden

help arith

muestra una lista con los operadores aritméticos que pueden ser empleados en MATLAB.
Los nombres de las variables comienzan con una letra seguida de cualquier número de letras, dígitos o guiones bajos _ . Sin embargo, MATLAB solo utiliza los 31 primeros caracteres del nombre de la variable.
Un % indica el inicio de un comentario.
A continuación, el contenido de un archivo diary que muestra los resultados de algunas de las órdenes de MATLAB arriba descritas:

```
% Este un archivo diary
%

help diary

 DIARY Save text of MATLAB session.
   DIARY filename causes a copy of all subsequent command window input
   and most of the resulting command window output to be appended to the
   named file. If no file is specified, the file 'diary' is used.

   DIARY OFF suspends it.
   DIARY ON turns it back on.
   DIARY, by itself, toggles the diary state.

   Use the functional form of DIARY, such as DIARY('file'),
   when the file name is stored in a string.

version

ans =

6.5.0.180913a (R13)  % por omision, el resultado se guardado en una variable llamada ans

ver
-----------------------------------------------------------------------------
MATLAB Version 6.5.0.180913a (R13)
MATLAB License Number: 0
Operating System: Microsoft Windows XP Version 5.1 (Build 2600: Service Pack 1)
Java VM Version: Java 1.3.1_01 with Sun Microsystems Inc. Java HotSpot(TM) Client VM
-----------------------------------------------------------------------------
MATLAB                                Version 6.5     (R13)
Curve Fitting Toolbox                 Version 1.1     (R13)
MATLAB Compiler                       Version 3.0     (R13)
MATLAB Report Generator               Version 1.3     (R13)
```

```
Optimization Toolbox                 Version 2.2     (R13)
Partial Differential Equation Toolbox Version 1.0.4  (R13)
Spline Toolbox                       Version 3.1.1   (R13)
Statistics Toolbox                   Version 4.0     (R13)
Symbolic Math Toolbox                Version 2.1.3   (R13)
Wavelet Toolbox                      Version 2.2     (R13)
```

help ver

```
 VER MATLAB, Simulink and toolbox version information.
   VER displays the current MATLAB, Simulink and toolbox version information.
   VER(TOOLBOX_DIR) displays the current version information for the
   toolbox specified by the string TOOLBOX_DIR.
   A = VER displays the general MATLAB version header and return in A the
   sorted struct array of version information on all toolboxes on the
   MATLAB path.
   The definition of struct A is:
        A.Name     : toolbox name
        A.Version  : toolbox version number
        A.Release  : toolbox release string
        A.Date     : toolbox release date
   For example,
     ver control
    displays the version info for the Control System Toolbox, sorted
    alphabetically.
     A = ver('control');
    returns in A the version information for the Control System Toolbox,
    sorted alphabetically.

   For tips on how to get VER to display version information about
   your toolbox, type at the MATLAB prompt
      more on
      type ver.m
   and then type 'more off' when the display of ver.m has finished.

   See also VERSION, HOSTID, LICENSE, INFO, WHATSNEW.
```

lookfor ver

```
ISPC True for the PC (Windows) version of MATLAB.
ISUNIX True for the UNIX version of MATLAB.
ISVMS  True for the VMS version of MATLAB.
VER MATLAB, Simulink and toolbox version information.
VERTCAT Vertical concatenation.
BUILTIN  Execute built-in function from overloaded method.
```

```
SWITCH Switch among several cases based on expression.
INVHILB Inverse Hilbert matrix.
IPERMUTE Inverse permute array dimensions.
LOGICAL Convert numeric values to logical.
ACOS  Inverse cosine.

%  Con Ctrl+C se interrumpe la ejecucion de una orden

date

ans =

04-Dec-2005

clock

ans =

  1.0e+003 *

    2.0050    0.0120    0.0040    0.0170    0.0590    0.0292

help clock

 CLOCK  Current date and time as date vector.
    CLOCK returns a six element date vector vector containing the
    current time and date in decimal form:

       CLOCK = [year month day hour minute seconds]

    The first five elements are integers. The seconds element
    is accurate to several digits beyond the decimal point.
    FIX(CLOCK) rounds to integer display format.

    See also DATEVEC, DATENUM, NOW, ETIME, TIC, TOC, CPUTIME.

!dir ar*.*              % ejecucion de una orden del sistema operativo
 El volumen de la unidad C no tiene etiqueta.
 El número de serie del volumen es: 0000-25C4

 Directorio de C:\MATLAB6p5\work

08/07/2005  05:34 p.m.          16,850 archivos
07/04/2005  03:18 p.m.             382 areas.m
```

```
07/04/2005  01:14 a.m.               372 area_circulo.m
               3 archivos         17,604 bytes
               0 dirs  19,754,225,664 bytes libres
pwd

ans =

C:\MATLAB6p5\work

help arith

 Arithmetic operators.
 +   Plus.
     X + Y adds matrices X and Y.  X and Y must have the same
     dimensions unless one is a scalar (a 1-by-1 matrix).
     A scalar can be added to anything.

 -   Minus.
     X - Y subtracts matrix X from Y.  X and Y must have the same
     dimensions unless one is a scalar.  A scalar can be subtracted
     from anything.

 *   Matrix multiplication.
     X*Y is the matrix product of X and Y.  Any scalar (a 1-by-1 matrix)
     may multiply anything.  Otherwise, the number of columns of X must
     equal the number of rows of Y.

 .*  Array multiplication
     X.*Y denotes element-by-element multiplication.  X and Y
     must have the same dimensions unless one is a scalar.
     A scalar can be multiplied into anything.

 ^   Matrix power.
     Z = X^y is X to the y power if y is a scalar and X is square. If y is an
     integer greater than one, the power is computed by repeated
     multiplication. For other values of y the calculation
     involves eigenvalues and eigenvectors.
     Z = x^Y is x to the Y power, if Y is a square matrix and x is a scalar,
     computed using eigenvalues and eigenvectors.
     Z = X^Y, where both X and Y are matrices, is an error.

 .^  Array power.
     Z = X.^Y denotes element-by-element powers.  X and Y
     must have the same dimensions unless one is a scalar.
```

A scalar can operate into anything.

```
a=3

a =

   3

b=4;   % el resultado no es mostrado al terminar una orden con ;

c=sqrt(a^2+b^2)

c =

   5

clc   %-se borra la ventana de ordenes

quit
```

2.3.- Manejo del área de trabajo

Las variables y arreglos definidos en MATLAB son almacenados en una parte de la memoria de la computadora conocida como *área de trabajo(workspace)*.

who	Muestra los nombres de las variables del área de trabajo
whos	Despliega los nombres de las variables del área de trabajo, su tamaño, el número de bytes empleados y su tipo
clear	Elimina variables del área de trabajo
save archivo	Guarda el contenido del área de trabajo en un archivo
load archivo	Carga en el área de trabajo la información guardada previamente con save
dir	Muestra el contenido del directorio actual de trabajo
ls	Equivalente a dir
delete(cadena)	Elimina al archivo indicado por una cadena de caracteres

disp	Muestra el valor de una variables
format	Especifica cual es el formato con que será impresa la información
fprintf	Genera una salida impresa de acuerdo a un formato especificado por el usuario

El siguiente es el contenido de un archivo diary con los resultados de algunos de los comandos arriba indicados:

```
% Este es un archivo diary
%

% x es un vector cuyos valores son 0, .1, .2, .3, .4, .5, .6, .7, .8, .9 y 1.0
x=0:.1:1

x =

 Columns 1 through 6

      0    0.1000    0.2000    0.3000    0.4000    0.5000

 Columns 7 through 11

   0.6000    0.7000    0.8000    0.9000    1.0000

y=sin(x)        % calculo de sin (x(1)), sin(x(2)), sin(x(3)), ..., sin(x(11))

y =

 Columns 1 through 6

      0    0.0998    0.1987    0.2955    0.3894    0.4794

 Columns 7 through 11

   0.5646    0.6442    0.7174    0.7833    0.8415


% multiplicacion de los vectores x y y, elemento por elemento:
% z(1) = x(1)*y(1)
% z(2) = x(2)*y(2)
% z(3) = x(3)*y(3) .....
```

```
z=x.*y

z =

 Columns 1 through 6

      0   0.0100   0.0397   0.0887   0.1558   0.2397

 Columns 7 through 11

   0.3388   0.4510   0.5739   0.7050   0.8415

f=sqrt(-5)

f =

      0 + 2.2361i

g=sin(f)/(4.6-1.8i)   % i es la raiz de -1

g =

  -0.3412 + 0.8719i

who

Your variables are:

f g x y z

whos
  Name      Size              Bytes  Class

  f         1x1                  16  double array (complex)
  g         1x1                  16  double array (complex)
  x         1x11                 88  double array
  y         1x11                 88  double array
  z         1x11                 88  double array

Grand total is 35 elements using 296 bytes

who f

Your variables are:
```

f

whos x
 Name Size Bytes Class

 x 1x11 88 double array

Grand total is 11 elements using 88 bytes

save vars

% mat es la extension o sufijo asignada a los archivos que contienen copias de espacios de
% trabajo

ls vars.mat

vars.mat

dir vars.mat

vars.mat

clear f

clear g x

who

Your variables are:

y z

clear

who

load vars

who

Your variables are:

f g x y z

```
clear

delete('vars.mat')

ls vars.mat

vars.mat not found.

help format

 FORMAT Set output format.
   All computations in MATLAB are done in double precision.
   FORMAT may be used to switch between different output
   display formats as follows:
    FORMAT       Default. Same as SHORT.
    FORMAT SHORT   Scaled fixed point format with 5 digits.
    FORMAT LONG    Scaled fixed point format with 15 digits.
    FORMAT SHORT E Floating point format with 5 digits.
    FORMAT LONG E  Floating point format with 15 digits.
    FORMAT SHORT G Best of fixed or floating point format with 5 digits.
    FORMAT LONG G  Best of fixed or floating point format with 15 digits.
    FORMAT HEX     Hexadecimal format.
    FORMAT +       The symbols +, - and blank are printed
                   for positive, negative and zero elements.
                   Imaginary parts are ignored.
    FORMAT BANK    Fixed format for dollars and cents.
    FORMAT RAT     Approximation by ratio of small integers.

   Spacing:
    FORMAT COMPACT Suppress extra line-feeds.
    FORMAT LOOSE   Puts the extra line-feeds back in.

R

R =

   8.3140

format long, R

R =

  8.31400000000000
```

```
format short e

R

R =

 8.3140e+000

format long e, R

R =

   8.314000000000000e+000

format   % el default es short

R

R =

   8.3140


n=50. R=8.314

n =

   50


R =

   8.3140

disp(n), disp(R)
   50

   8.3140

fprintf('%f  %d',R,n)
8.314000  50

fprintf('R=%8.3f J/mol K     n=%3d',R,n)
```

R= 8.314 J/mol K n= 50

fprintf(' R=%8.3f J/mol K \n n=%3d',R.n)
R= 8.314 J/mol K
n= 50

3.- Gráficas en 2D

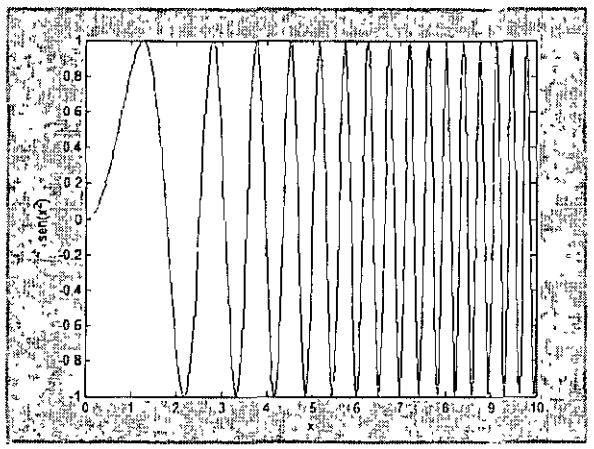
3.1.- Gráficas de funciones indicadas en una cadena de caracteres

Los gráficos resultantes de la ejecución de órdenes tales como fplot o plot son mostrados en *figuras*. A menos que se indique otra situación, la figura empleada es la número 1 (Figure No. 1) que entonces se convierte en la *figura actual* (*current figure*). Todas las órdenes relacionadas con la creación y/o modificación de gráficos se aplican siempre a la figura actual.

fplot	Gráfica de una función especificada como una cadena de caracteres
xlabel	Colocación de una cadena de caracteres como etiqueta del eje horizontal x
ylabel	Colocación de una cadena de caracteres como etiqueta del eje vertical y
figure	Apertura de una figura y su conversión en figura actual

Un archivo diary que muestra los resultados de algunas de las órdenes arriba indicadas es:

```
% Este es un archivo diary
%
% un . antes de *, / o ^ indica operaciones vectoriales elemento por elemento

fplot('sin(x.^2)', [0,10])          % grafica de sen(x^2),  0<=x<=10

xlabel('x'), ylabel('sen(x^2)'), figure(1)
```



3.2.- Gráficas de vectores

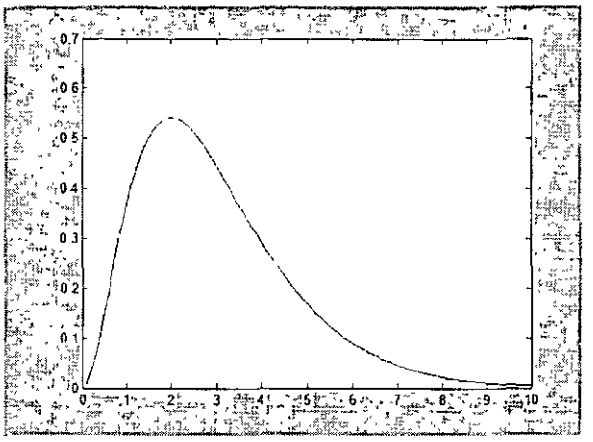
La orden plot permite graficar en una forma muy eficiente los valores de vectores generados dentro de MATLAB.

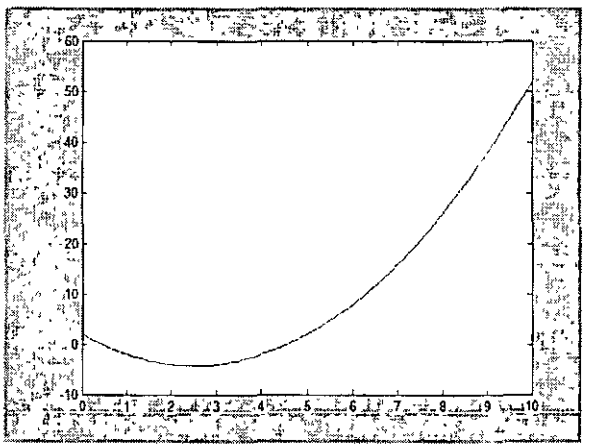
plot	Genera la gráfica de un vector contra otro
hold	Permite o no que una gráfica sea agregada a las anteriores
title	Coloca un título a la figura actual
legend	Genera un cuadro que permite identificar a las curvas mostradas en la figura
text	Inserta un texto en una figura
grid	Dibuja un mallado
clf	Borrado del contenido de una figura
ginput	Obtiene las coordenadas de puntos dentro de una figura

En seguida, un archivo diary con las órdenes empleadas para generar algunas gráficas de vectores.

```
% Este es un archivo diary
%
figure(1)

clf

x=0:.1:10;
y=x.^2.*exp(-x);   % y=x^2*exp(-x)

plot(x,y)

plot(x, x.^2-5*x+2)   % esta grafica sustituye a la anterior en la figura 1
```

Las gráficas construidas con estas órdenes son:





Otros archivos diary diseñados para mostrar algunas órdenes plot:

```
% Este es un archivo diary
%

help plot

 PLOT   Linear plot.
    PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
    then the vector is plotted versus the rows or columns of the matrix,
    whichever line up.  If X is a scalar and Y is a vector, length(Y)
    disconnected points are plotted.

    PLOT(Y) plots the columns of Y versus their index.
```

If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)). In all other uses of PLOT, the imaginary part is ignored.

Various line types, plot symbols and colors may be obtained with PLOT(X,Y,S) where S is a character string made from one element from any or all the following 3 columns:

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star		
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus at each data point; PLOT(X,Y,'bd') plots blue diamond at each data point but does not draw any line.

PLOT(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combines the plots defined by the (X,Y,S) triples, where the X's and Y's are vectors or matrices and the S's are strings.

For example, PLOT(X,Y,'y-',X,Y,'go') plots the data twice, with a solid yellow line interpolating green circles at the data points.

The PLOT command, if no color is specified, makes automatic use of the colors specified by the axes ColorOrder property. The default ColorOrder is listed in the table above for color systems where the default is blue for one line, and for multiple lines, to cycle through the first six colors in the table. For monochrome systems, PLOT cycles over the axes LineStyleOrder property.

PLOT returns a column vector of handles to LINE objects, one handle per line.

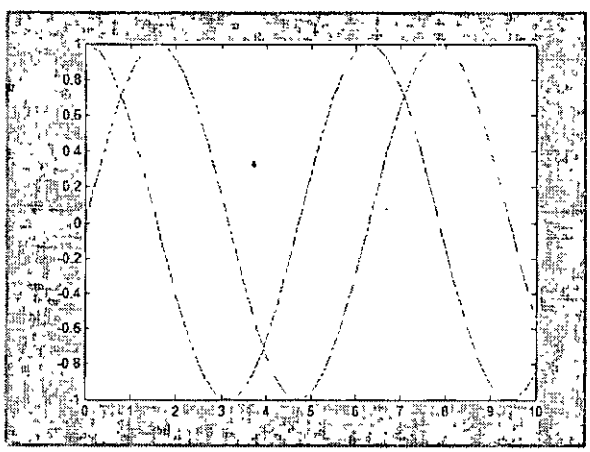
The X,Y pairs, or X,Y,S triples, can be followed by parameter/value pairs to specify additional properties

of the lines.

See also SEMILOGX, SEMILOGY, LOGLOG, PLOTYY, GRID, CLF, CLC, TITLE, XLABEL, YLABEL, AXIS, AXES, HOLD, COLORDEF, LEGEND, SUBPLOT, STEM.

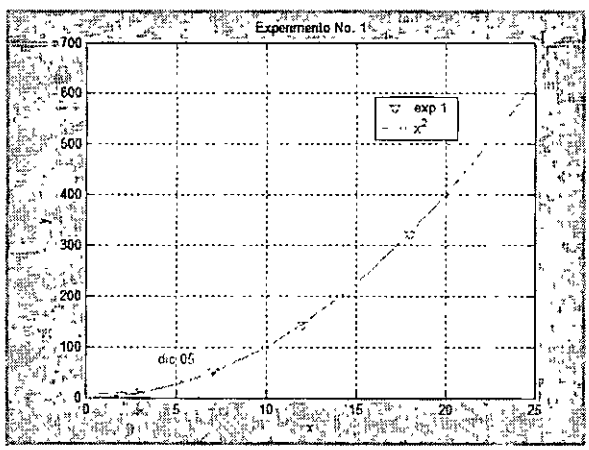
Overloaded methods
help cfit/plot.m
help ntree/plot.m
help dtree/plot.m
help wvtree/plot.m
help rwvtree/plot.m
help edwttree/plot.m

```
figure(2)          % la figura 2 se convierte en la actual

hold on            % permite juntar varias graficas en una figura

plot(x,sin(x),'g')         % color verde

plot(x,cos(x),'m')         % color magenta
```



```
% Este es un archivo diary
%
clear

figure(1),clf

% a los vectores x y y, se le asignan directamente valores
x=[0,1,2,3,5,7,10,12,15,18,20,23,25];
```

```
y=[0..98,2.1,8.78,24.97,49.5.101,142.9,226....
   321,402.531,631]          % ... indican que la orden continua en la linea de abajo

y =

 Columns 1 through 7

      0   0.9800   2.1000   8.7800  24.9700  49.5000 101.0000

 Columns 8 through 13

 142.9000 226.0000 321.0000 402.0000 531.0000 631.0000

plot(x,y,'vc',x,x.^2,'g')

xlabel('x'), ylabel('y')

title('Experimento No. 1')     % adicion de un titulo a la grafica

legend('exp 1','x^2')          % generacion de un cuadro para identificar las curvas

text(4,75),'dic 05')           % añadido de un texto en las coordenadas (4,75)

grid
```



La orden ginput permite obtener las coordenadas de un punto cualquiera dentro de una figura. Por ejemplo, después de ejecutar

[xcord, ycord] = ginput

manualmente es señalado un punto dentro de la figura con el ratón. Al presionar el botón izquierdo entonces las coordenadas del punto señalado son guardadas en los arreglos xcord y ycord. Este procedimiento es repetido cuantas veces se requiera y puede ser finalizado al presionar la tecla `Enter`. Con la orden

[xcord, ycord] = ginput(3)

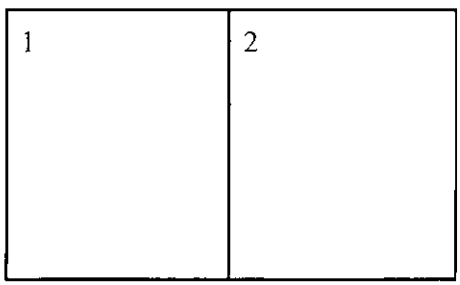
se especifica el número de puntos cuyas coordenadas serán almacenadas en los arreglos xcord y y ycord.

3.3.- Varias gráficas en una misma figuras

Dentro de una misma figura es posible representar más de una gráfica. Con la orden subplot, una figura es dividida en varias regiones y en cada una de ellas se puede colocar una gráfica distinta. Por ejemplo; con

subplot(2,1,1)

una figura es dividida en dos partes, como si fuera un arreglo de 2 filas y 1columna. La parte superior es numerada como 1 y la inferior como 2. Esta misma orden hace que la parte 1 de la figura sea ahora la figura actual.



Por otra parte, la orden

subplot(2,1,2)

divide la figura en dos regiones organizadas como un arreglo de 2 filas y 1columna. La parte superior es numerada como 1 y la inferior como 2. La parte 2 queda como figura actual.

1
2

linspace	Crea un vector de valores uniformemente espaciados
subplot	Divide una figura en varias regiones
semilogx	Genera una gráfica semilogarítmica (eje horizontal logarítmico, eje vertical lineal)
semilogy	Genera una gráfica semilogarítmica (eje horizontal lineal, eje vertical logarítmico)
loglog	Genera una gráfica logarítmica(ambos ejes logarítmicos)
polar	Graficación en coordenadas polares
pol2cart	Conversión de coordenadas polares a cartesianas

Ahora, un archivo diary que ilustra el uso de las órdenes anteriores.

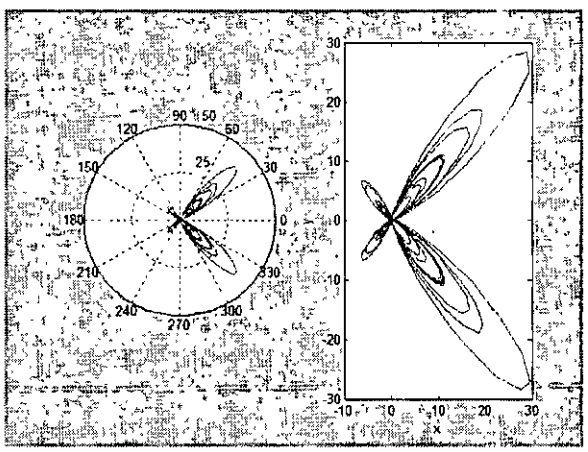
```
% Este es un archivo diary
%

figure(1)

help linspace

 LINSPACE Linearly spaced vector.
   LINSPACE(X1, X2) generates a row vector of 100 linearly
   equally spaced points between X1 and X2.

   LINSPACE(X1, X2, N) generates N points between X1 and X2.
   For N < 2, LINSPACE returns X2.
```

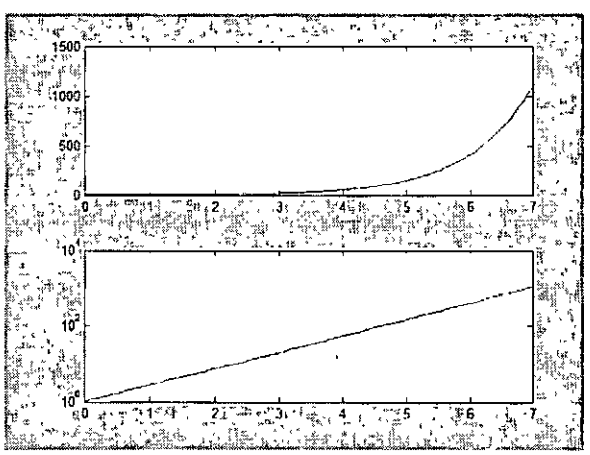
```
   See also LOGSPACE, :.

x=linspace(0,7);  % por omision, se generan 100 valores uniformemente espaciados

whos x
  Name      Size              Bytes  Class

  x         1x100               800  double array

Grand total is 100 elements using 800 bytes

y=exp(x);

subplot(2,1,1)  % grafica 1 en un arreglo de 2 filas y 1 columna

plot(x,y)

subplot(2,1,2)  % grafica 2 en un arreglo de 2 filas y 1 columna

help semilogy

 SEMILOGY Semi-log scale plot.
   SEMILOGY(...) is the same as PLOT(...), except a
   logarithmic (base 10) scale is used for the Y-axis.

   See also PLOT.

semilogy(x,y)

figure(2)

t=linspace(0,22*pi,1100);      % pi es la constante 3.141592.....

r=exp(cos(t)-2*cos(4*t)+(sin(t/12)).^5);        % r=exp(cos(t)-2cos(4t)+sin(t/12)^.5)

subplot(1,2,1)  % grafica 1 en un arreglo de 1 filas y 2 columnas

help polar

 POLAR  Polar coordinate plot.
   POLAR(THETA, RHO) makes a plot using polar coordinates of
   the angle THETA, in radians, versus the radius RHO.
   POLAR(THETA,RHO,S) uses the linestyle specified in string S.
   See PLOT for a description of legal linestyles.
```

See also PLOT. LOGLOG. SEMILOGX. SEMILOGY.

```
polar(t.r)
subplot(1.2.2)   % grafica 2 en un arreglo de 1 fila y 2 columnas
[x,y]=pol2cart(t.r);          % conversion de coordenadas polares a cartesianas
plot(x,y)
xlabel('x'), ylabel('y ')
```

Las figuras 2 y 1 obtenidas arriba son mostradas a continuación





3.4.- Respaldo de gráficas en un archivo

Las siguientes órdenes se usan para salvar figuras en un archivo en disco:

saveas	Guarda la figura actual en un archivo
openfig	Apertura de archivo que contiene una figura

La orden

```
saveas(gcf,'grafica1','fig')
```

guarda la figura actual en el archivo de nombre grafica.fig. Un archivo que contiene una figura debe tener la extensión o sufijo fig.
Para recuperar una figura almacenada en disco se emplea la orden openfig:

```
openfig('grafica1')
```

en la cual se especifica como una cadena de caracteres el nombre del archivo. Si se omite la extensión, se considera entonces que es fig. La figura abierta se convierte en la actual.
En el menú de una figura también es posible guardarla al seleccionar `File` y luego `Save As`.
También es posible abrir otra figura mediante la selección `File` seguida de `Open`.

4.- Matrices y álgebra lineal

4.1.- Vectores. Operaciones con vectores.

Algunas de las funciones más usadas en el manejo de vectores son:

dot	Producto punto de dos vectores
norm	Cálculo de la norma de un vector
cross	Producto vectorial de dos vectores de 3 componentes
size	Proporciona el tamaño de un arreglo
sum	Suma de los elementos de un vector
prod	Producto de los elementos de un vector
median	Mediana de un vector
mean	Media de un vector
var	Varianza de un vector
std	Desviación típica de un vector
sort	Ordenamiento en forma ascendente de un vector
max	Mayor elemento de un vector
min	Menor elemento de un vector

En seguida, un archivo diary con algunas de las órdenes indicadas arriba.

```
% Este es un archivo diary
%

a=[2,0,-4]

a =

   2   0   -4
```

```
b=[-1,3,5]

b =

   -1    3    5

3*a+2*b        % suma de vectores

ans =

    4    6   -2

a.*b     % multiplicacion elemento por elemento

ans =

   -2    0  -20

dot(a,b)   % producto punto o producto escalar

ans =

  -22

help norm

 NORM   Matrix or vector norm.
   For matrices...
     NORM(X) is the largest singular value of X, max(svd(X)).
     NORM(X,2) is the same as NORM(X).
     NORM(X,1) is the 1-norm of X, the largest column sum,
                 = max(sum(abs(X))).
     NORM(X,inf) is the infinity norm of X, the largest row sum,
                 = max(sum(abs(X'))).
     NORM(X,'fro') is the Frobenius norm, sqrt(sum(diag(X'*X))).
     NORM(X,P) is available for matrix X only if P is 1, 2, inf or 'fro'.

   For vectors...
     NORM(V,P) = sum(abs(V).^P)^(1/P).
     NORM(V) = norm(V,2).
     NORM(V,inf) = max(abs(V)).
     NORM(V,-inf) = min(abs(V)).

   See also COND, RCOND, CONDEST, NORMEST.
```

```
norm(b)

ans =

   5.9161

% calculo del angulo entre dos vectores (en radianes)
theta=acos(dot(a,b)/(norm(a)*norm(b)))

theta =

   2.5526

a_unitario=a/norm(a)    % calculo de un vector unitario

a_unitario =

   0.4472         0   -0.8944

norm(a_unitario)

ans =

    1

cross(a,b)        % producto cruz de a y b

ans =

   12    -6     6

vec=[0,3,5,0,-1,7,1.5,8,-4,11]

vec =

 Columns 1 through 8

       0    3.0000    5.0000         0   -1.0000    7.0000    1.5000    8.0000

 Columns 9 through 10

  -4.0000   11.0000

size(vec)         % tamaño de un vector
```

```
ans =

    1    10

[m,n]=size(vec)

m =

     1


n =

    10

% vector(subindice inicial : incremento : subindice final)

vec(1:2:n-1)     % vec(1),vec(3),vec(5),...,vec(n-1)

ans =

         0    5.0000   -1.0000    1.5000   -4.0000

vec(n:-1:5)      % vec(n),vec(n-1),vec(n-2),...,vec(5)

ans =

   11.0000   -4.0000    8.0000    1.5000    7.0000   -1.0000

sum(vec)         % vec(1)+vec(2)+vec(3)+...+vec(n)

ans =

   30.5000

prod(vec(5:10))          % vec(5)*vec(6)*vec(7)*vec(8)*vec(9)*vec(10)

ans =

       3696

mean(vec)

ans =
```

```
   3.0500

median(vec)

ans =

   2.2500

sort(vec)  % ordenamiento en forma ascendente

ans =

 Columns 1 through 8

  -4.0000   -1.0000         0         0    1.5000    3.0000    5.0000    7.0000

 Columns 9 through 10

   8.0000   11.0000

var(vec), std(vec)   % varianza y desviación típica

ans =

  21.5806


ans =

   4.6455

max(vec), min(vec)

ans =

   11


ans =

   -4
```

4.2.- Matrices. Operaciones con matrices

Algunas de las funciones más comunes en el cálculo matricial son:

eye	Generación de una matrız identidad
det	Determinante de una matriz
inv	Inversa de una matrız
rank	Rango de una matriz
svd	Descomposición en valores singulares
cond	Número de condición de una matriz
lu	Factorización LU
eig	Valores propios de una matriz

El siguiente es un archivo diary que ilustra las órdenes arriba mencionadas:

```
% Este es un archivo diary
%

A=[3,6,-5;7,10,0;-1,2,9]   % asignacion de valores a una matriz

A =

   3    6   -5
   7   10    0
  -1    2    9

whos A
  Name      Size                Bytes  Class

  A         3x3                    72  double array

Grand total is 9 elements using 72 bytes

size(A)

ans =

   3    3
```

```
[m.n]=size(A)

m =

    3


n =

    3

A(:,2)  % segunda columna de A

ans =

    6
   10
    2

A(1,:)   % primera fila de A

ans =

    3    6   -5

At=A'    % ' es el operador de transposicion de matrices

At =

    3    7   -1
    6   10    2
   -5    0    9

B=[5,8,3;-4,5,0;10,1,-7]

B =

    5    8    3
   -4    5    0
   10    1   -7

A+B              % suma de matrices

ans =
```

```
    8.  14   -2
    3   15    0
    9    3    2

5*A-2*B

ans =

     5   14  -31
    43   40    0
   -25    8   59

A.*B            % multiplicacion elemento por elemento

ans =

    15   48  -15
   -28   50    0
   -10    2  -63

A*B             % producto matricial

ans =

   -59   49   44
    -5  106   21
    77   11  -66

eye(2)          % matriz identidad de orden 2

ans =

     1    0
     0    1

3*A^2-4*A+eye(3)     % 3*A^2-4*A+I

ans =

   157   180  -160
   245   387  -105
    10    88   223

det(A)
```

```
ans =

 -228

inv(A)  % inversa de A

ans =

  -0.3947   0.2807  -0.2193
   0.2763  -0.0965   0.1535
  -0.1053   0.0526   0.0526

inv(A)*A

ans =

   1.0000   0.0000   0.0000
   0.0000   1.0000   0.0000
  -0.0000   0.0000   1.0000

A^-1

ans =

  -0.3947   0.2807  -0.2193
   0.2763  -0.0965   0.1535
  -0.1053   0.0526   0.0526

A^-2

ans =

   0.2565  -0.1494   0.1181
  -0.1519   0.0950  -0.0673
   0.0506  -0.0319   0.0339

inv(A)*inv(A)

ans =

   0.2565  -0.1494   0.1181
  -0.1519   0.0950  -0.0673
   0.0506  -0.0319   0.0339
```

```
rank(A)          % rango de A

ans =

   3

max(A)

ans =

   7   10    9

min(A)

ans =

  -1    2   -5

help norm

 NORM   Matrix or vector norm.
   For matrices...
    NORM(X) is the largest singular value of X, max(svd(X)).
    NORM(X,2) is the same as NORM(X).
    NORM(X,1) is the 1-norm of X, the largest column sum,
              = max(sum(abs(X))).
    NORM(X,inf) is the infinity norm of X, the largest row sum,
              = max(sum(abs(X'))).
    NORM(X,'fro') is the Frobenius norm, sqrt(sum(diag(X'*X))).
    NORM(X,P) is available for matrix X only if P is 1, 2, inf or 'fro'.

   For vectors...
    NORM(V,P) = sum(abs(V).^P)^(1/P).
    NORM(V) = norm(V,2).
    NORM(V,inf) = max(abs(V)).
    NORM(V,-inf) = min(abs(V)).

   See also COND, RCOND, CONDEST, NORMEST.

norm(A)

ans =

  14.1502
```

```
norm(A,2)

ans =

  14.1502

norm(A,'inf')

ans =

   17
norm(A,'fro')

ans =

  17.4642

help cond

 COND   Condition number with respect to inversion.
   COND(X) returns the 2-norm condition number (the ratio of the
   largest singular value of X to the smallest). Large condition
   numbers indicate a nearly singular matrix.

   COND(X,P) returns the condition number of X in P-norm:

     NORM(X,P) * NORM(INV(X),P).

   where P = 1, 2, inf, or 'fro'.

   See also RCOND, CONDEST, CONDEIG, NORM, NORMEST.

cond(A)

ans =

   8.8794

help svd

 SVD    Singular value decomposition.
   [U,S,V] = SVD(X) produces a diagonal matrix S, of the same
   dimension as X and with nonnegative diagonal elements in
   decreasing order, and unitary matrices U and V so that
```

```
   X = U*S*V'.

   S = SVD(X) returns a vector containing the singular values.

   [U,S,V] = SVD(X,0) produces the "economy size"
   decomposition. If X is m-by-n with m > n, then only the
   first n columns of U are computed and S is n-by-n.

   See also SVDS, GSVD.

 Overloaded methods
   help sym/svd.m

svd(A)

ans =

   14.1502
   10.1111
    1.5936

help lu

 LU    LU factorization.
   [L,U] = LU(X) stores an upper triangular matrix in U and a
   "psychologically lower triangular matrix" (i.e. a product
   of lower triangular and permutation matrices) in L, so
   that X = L*U. X can be rectangular.

   [L,U,P] = LU(X) returns unit lower triangular matrix L, upper
   triangular matrix U, and permutation matrix P so that
   P*X = L*U.

   Y = LU(X) returns the output from LAPACK'S DGETRF or ZGETRF
   routine if X is full. If X is sparse, Y contains the strict
   lower triangle of L embedded in the same matrix as the upper
   triangle of U. In both full and sparse cases, the permutation
   information is lost.

   [L,U,P,Q] = LU(X) returns unit lower triangular matrix L,
   upper triangular matrix U, a permutation matrix P and a column
   reordering matrix Q so that P*X*Q = L*U for sparse non-empty X.
   This uses UMFPACK and is significantly more time and memory
   efficient than the other syntaxes, even when used with COLAMD.
```

[L,U,P] = LU(X,THRESH) controls pivoting in sparse matrices, where THRESH is a pivot threshold in [0,1]. Pivoting occurs when the diagonal entry in a column has magnitude less than THRESH times the magnitude of any sub-diagonal entry in that column. THRESH = 0 forces diagonal pivoting. THRESH = 1 is the default.

[L,U,P,Q] = LU(X,THRESH) controls pivoting in UMFPACK, where THRESH is a pivot threshold in [0,1]. Given a pivot column j, UMFPACK selects the sparsest candidate pivot row i such that the absolute value of the pivot entry is greater than or equal to THRESH times the largest entry in the column j. The magnitude of entries in L is limited to 1/THRESH. A value of 1.0 results in conventional partial pivoting. The default value is 0.1. Smaller values tend to lead to sparser LU factors, but the solution can become inaccurate. Larger values can lead to a more accurate solution (but not always), and usually an increase in the total work.

See also COLAMD, LUINC, QR, RREF, UMFPACK.

```
[L,U,P]=lu(A)

L =

   1.0000         0         0
  -0.1429    1.0000         0
   0.4286    0.5000    1.0000


U =

   7.0000   10.0000         0
         0    3.4286    9.0000
         0         0   -9.5000


P =

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

```
P*A

ans =

   7   10    0
  -1    2    9
   3    6   -5

L*U

ans =

   7.0000   10.0000         0
  -1.0000    2.0000    9.0000
   3.0000    6.0000   -5.0000

P*A==L*U

ans =

   1   1   1
   1   1   1
   1   1   0

help eig

 EIG   Eigenvalues and eigenvectors.
   E = EIG(X) is a vector containing the eigenvalues of a square
   matrix X.

   [V,D] = EIG(X) produces a diagonal matrix D of eigenvalues and a
   full matrix V whose columns are the corresponding eigenvectors so
   that X*V = V*D.

   [V,D] = EIG(X,'nobalance') performs the computation with balancing
   disabled, which sometimes gives more accurate results for certain
   problems with unusual scaling. If X is symmetric, EIG(X,'nobalance')
   is ignored since X is already balanced.

   E = EIG(A,B) is a vector containing the generalized eigenvalues
   of square matrices A and B.

   [V,D] = EIG(A,B) produces a diagonal matrix D of generalized
   eigenvalues and a full matrix V whose columns are the
```

```
   corresponding eigenvectors so that A*V = B*V*D.

   EIG(A,B,'chol') is the same as EIG(A,B) for symmetric A and symmetric
   positive definite B.  It computes the generalized eigenvalues of A and B
   using the Cholesky factorization of B.
   EIG(A,B,'qz') ignores the symmetry of A and B and uses the QZ algorithm.
   In general, the two algorithms return the same result, however using the
   QZ algorithm may be more stable for certain problems.
   The flag is ignored when A and B are not symmetric.

   See also CONDEIG, EIGS.

 Overloaded methods
   help sym/eig.m

eig(A)

ans =

  -1.6423
  12.7759
  10.8664

% V es una matriz cuyas columnas son los vectores propios de A
% D es una matriz diagonal con los valores propios de A
[V,D]=eig(A)

V =

   0.8438   -0.3429    0.0870
  -0.5074   -0.8646    0.7026
   0.1746   -0.3672    0.7063


D =

  -1.6423         0         0
        0   12.7759         0
        0         0   10.8664
```

5.- Sistemas de ecuaciones lineales

El operador \ permite resolver sistemas de ecuaciones lineales. Este operador funciona para cualquier número de ecuaciones y de incógnitas.
Si el número de ecuaciones y de incógnitas es el mismo, el operador \ resuelve el sistema mediante descomposición LU.
En el caso sobredeterminado , más ecuaciones que incógnitas, el sistema usualmente es inconsistente y por lo tanto no tiene una solución exacta y entonces el operador \ encuentra una solución aproximada que minimiza a la norma del vector $(Ax-b)^2$ (solución de mínimos cuadrados).
Si el sistema tiene menos ecuaciones que incógnitas, el sistema es subdeterminado y usualmente tiene un número infinito de soluciones. El operador \ calcula una de ellas. La función pinv también puede ser empleada en este caso.

pinv	Cálculo de la matriz pseudo-inversa de Moore-Penrose

Un archivo diary que ilustra el tópico de la solución de sistemas lineales es:

```
% Este es un archivo diary
%

%  4x1 + x2    -6x3 =  0
% 10x1          +3x3 = -5
%    x1 +6x2 +13x3 =  2

A=[4,1,-6;10,0,3;1,6,13]

A =

   4   1  -6
  10   0   3
   1   6  13

b=[0;-5;2]

b =

   0
  -5
   2

x=inv(A)*b    % solucion mediante el calculo de la matriz inversa de la matriz de coeficientes A
```

```
x =

  -0.4490
   0.7764
  -0.1699

x=A\b          % solucion mediante descomposicion LU

x =

  -0.4490
   0.7764
  -0.1699

% Solucion de un sistema sobredeterminado (mas ecuaciones que incognitas)

% x1 +  x2 = 2
% x1 + 2x2 = 0
% x1 + 3x2 =-1

A=[1,1;1,2;1,3]

A =

     1     1
     1     2
     1     3

b=[2;0;-1]

b =

     2
     0
    -1

x=A\b  % solucion de minimos cuadrados

x =

   3.3333
  -1.5000

% Solucion de un sistema subdeterminado (menos ecuaciones que incognitas)
```

```
% x1 +  x2 +  x3 = 2
% x1 + 2x2 + 3x3 = 3

A=[1,1,1;1,2,3],b=[2;3]

A =

     1     1     1
     1     2     3


b =

     2
     3

x=A\b   % calculo de una solucion entre las muchas posibles

x =

    1.5000
         0
    0.5000

x=pinv(A)*b  % se encuentra otra solucion a traves de la pseudoinversa de Moore-Penrose


x =

    1.1667
    0.6667
    0.1667

help pinv

 PINV   Pseudoinverse.
    X = PINV(A) produces a matrix X of the same dimensions
    as A' so that A*X*A = A, X*A*X = X and A*X and X*A
    are Hermitian. The computation is based on SVD(A) and any
    singular values less than a tolerance are treated as zero.
    The default tolerance is MAX(SIZE(A)) * NORM(A) * EPS.

    PINV(A,TOL) uses the tolerance TOL instead of the default.
```

See also RANK.

```
pinv(A)

ans =

    1.3333   -0.5000
    0.3333   -0.0000
   -0.6667    0.5000
```

6.- Gráficas en 3D

Las siguientes son funçiones útiles para generar gráficas en 3 dimensiones:

meshgrid	Preparan arreglos para gráficas en 3D
mesh	Gráfica tipo malla de alambre de una superficie
surf	Gráfica de una superficie
contour	Gráfica de contornos de una superficie
meshc	Gráficas tipo malla de alambre y de contornos de una superficie
surfc	Gráficas de superficie y de contornos
clabel	Etiquetado de las curvas de nivel producidas por contour
colorbar	Muestra una barra de colores
colormap	Especifica el mapa de color empleado

A continuación, un archivo diary que muestra algunos comandos para dibujar gráficas en 3D

```
% Este es un archivo diary
%

clear

x=-2:1:2          % -2<=x<=2

x =

  -2   -1    0    1    2

y=-1:1:2          % -1<=y<=2

y =

  -1    0    1    2

help meshgrid
```

MESHGRID X and Y arrays for 3-D plots.
  [X,Y] = MESHGRID(x,y) transforms the domain specified by vectors
  x and y into arrays X and Y that can be used for the evaluation
  of functions of two variables and 3-D surface plots.
  The rows of the output array X are copies of the vector x and
  the columns of the output array Y are copies of the vector y.

  [X,Y] = MESHGRID(x) is an abbreviation for [X,Y] = MESHGRID(x,x).
  [X,Y,Z] = MESHGRID(x,y,z) produces 3-D arrays that can be used to
  evaluate functions of three variables and 3-D volumetric plots.

  For example, to evaluate the function x*exp(-x^2-y^2) over the
  range -2 < x < 2, -2 < y < 2,

    [X,Y] = meshgrid(-2:.2:2, -2:.2:2);
    Z = X .* exp(-X.^2 - Y.^2);
    mesh(Z)

  MESHGRID is like NDGRID except that the order of the first two input
  and output arguments are switched (i.e., [X,Y,Z] = MESHGRID(x,y,z)
  produces the same result as [Y,X,Z] = NDGRID(y,x,z)). Because of
  this, MESHGRID is better suited to problems in cartesian space,
  while NDGRID is better suited to N-D problems that aren't spatially
  based. MESHGRID is also limited to 2-D or 3-D.

  See also SURF, SLICE, NDGRID.

```
% Obtencion de arreglos xg y yg a partir de los vectores x y y
% xg y yg seran usados para graficar una superficie

[xg,yg]=meshgrid(x,y)

xg =

   -2   -1    0    1    2
   -2   -1    0    1    2
   -2   -1    0    1    2
   -2   -1    0    1    2


yg =

   -1   -1   -1   -1   -1
    0    0    0    0    0
```

```
   1   1   1   1   1
   2   2   2   2   2

whos xg yg
  Name      Size                Bytes  Class

  xg        4x5                   160  double array
  yg        4x5                   160  double array

Grand total is 40 elements using 320 bytes

zg=1./(1+xg.^2+yg.^2)          % z=1/(1+x^2+y^2) es evaluada en xg y yg

zg =

    0.1667    0.3333    0.5000    0.3333    0.1667
    0.2000    0.5000    1.0000    0.5000    0.2000
    0.1667    0.3333    0.5000    0.3333    0.1667
    0.1111    0.1667    0.2000    0.1667    0.1111

figure(1)

clf

mesh(xg,yg,zg)          % grafica tipo malla de alambre

% ahora, la misma grafica pero con mayor resolucion

figure(2),clf

x=-2:0.1:2;
y=-1:0.1:2;
[xg,yg]=meshgrid(x,y);
zg=1./(1+xg.^2+yg.^2);
mesh(xg,yg,zg)
title('z=1/(1+x^2+y^2)')
xlabel('x'),ylabel('y'),zlabel('z')

figure(3),clf

surf(xg,yg,zg)          % grafica de una superficie

xlabel('x'),ylabel('y'),zlabel('z'),title('z=1/(1+x^2+y^2)')
```

```
figure(4),clf

contour(xg,yg,zg)              % grafica de contornos

xlabel('x'),ylabel('y')
title('z=1/(1+x^2+y^2)')

figure(5),clf

contour(xg,yg,zg,3)   % 3 es el numero de contornos por generar (el valor por omision es 10)

figure(6),clf

c=contour(xg,yg,zg);
clabel(c)                      % etiquetado de las curvas de nivel
xlabel('x'),ylabel('y')
title('z=1/(x^2+y^2)')

figure(7),clf

c=contour(xg,yg,zg);

% con 'manual', el usuario selecciona con un click en el botón izquierdo del ratón las curvas
% que desea etiquetar. Al presionar la tecla Enter, el proceso termina.
clabel(c,'manual')

   Please wait a moment...

  Carefully select contours for labeling.
  When done, press RETURN while the Graph window is the active window.

figure(8),clf

h=contour(xg,yg,zg);

% ahora se especifican explicitamente los valores de z
clabel(h,[.3,.5,.8])

figure(9),clf

meshc(xg,yg,zg)          % graficas de malla de alambre y de contornos en una misma figura
xlabel('x'),ylabel('y'),zlabel('z')
title('z=1/(1+x^2+y^2)')
```

figure(10),clf

surfc(xg,yg,zg) % graficas de superficie y de contornos en una misma figura
xlabel('x'),ylabel('y'),zlabel('z')
title('z=1/(1+x^2+y^2)')

Las diez figuras generadas por las órdenes anteriores son mostradas a continuación:

figura 1

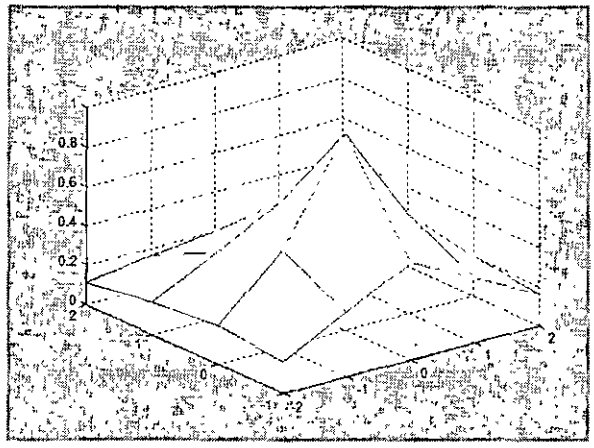


figura 2

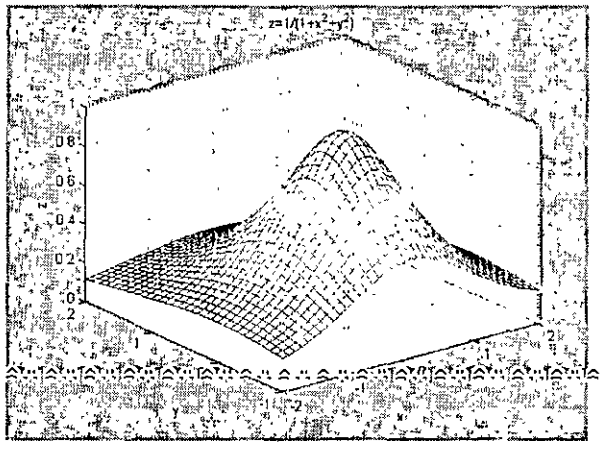


figura 3

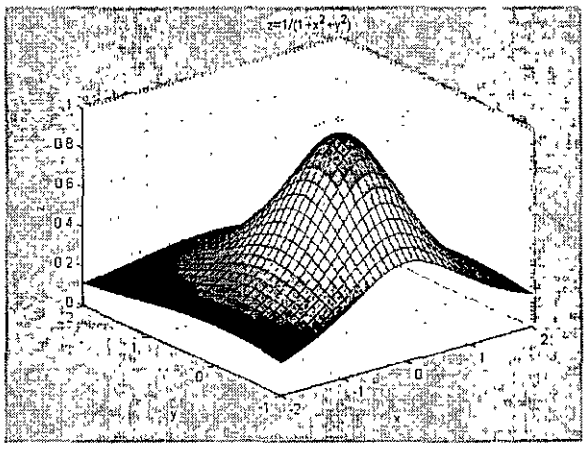


figura 4

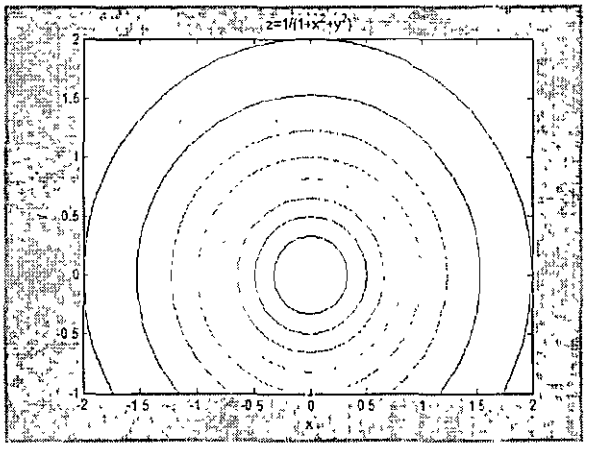


figura 5

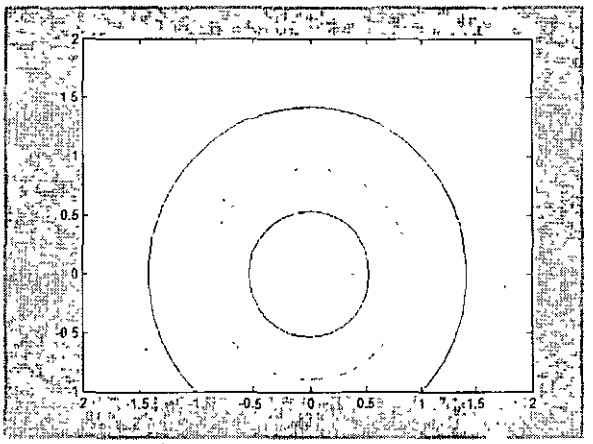


figura 6

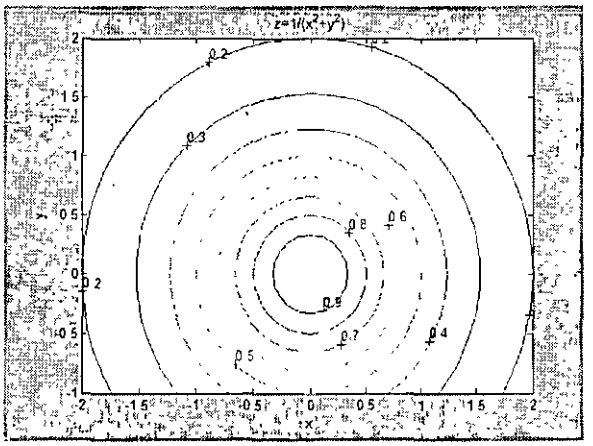


figura 7

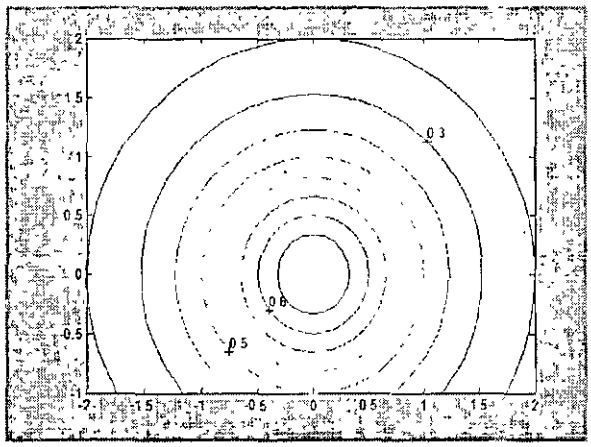


figura 8

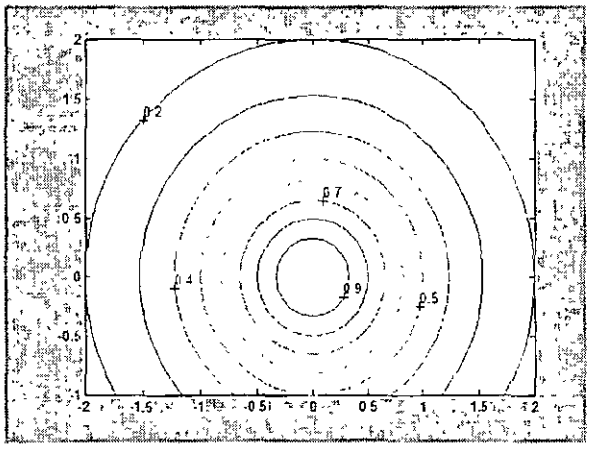


figura 9

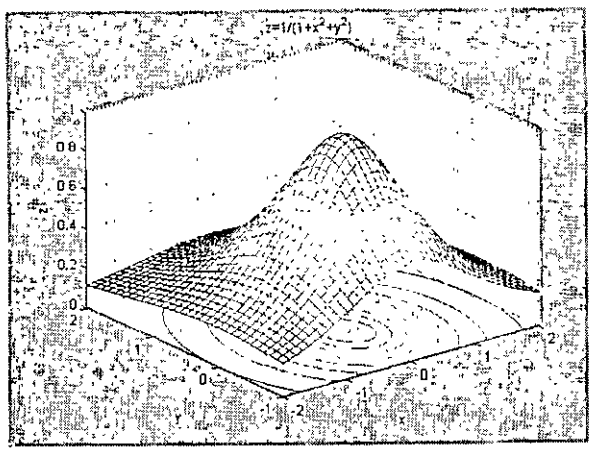
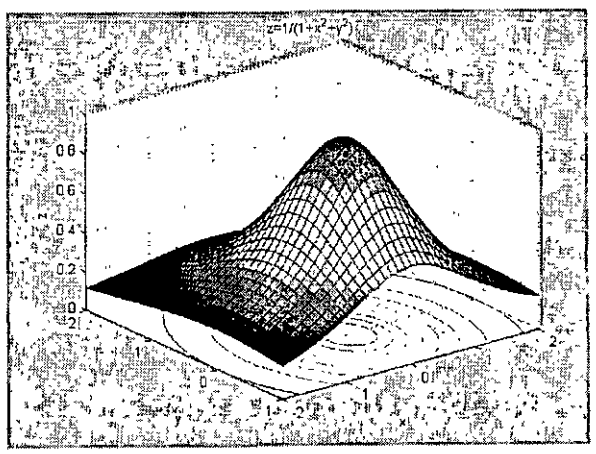


figura 10



La orden meshgrid no es la única forma de generar gráficas en 3D. Por ejemplo, para dibujar la superficie z = (2+sin(x))*cos(2y) en la región definida por $-2 \leq x \leq 2$ y $-3 \leq y \leq 3$ se puede emplear la secuencia de órdenes:

```
x=-2:0.1:2;
y=-3:0.1:3;

size(x);
m=ans(2);
size(y);
n=ans(2);
```

para generar un vector x de m elementos y otro vector y de n elementos. Entonces con dos ciclos for anidados se genera una matriz llamada z. El elemento z(i,j) contiene a la función por graficar evaluada en x(i) y y(j).

```
for i=1:m
  for j=1:n
     z(i,j)=(2+sin(x(i)))*cos(2*y(j));
  end
end
```

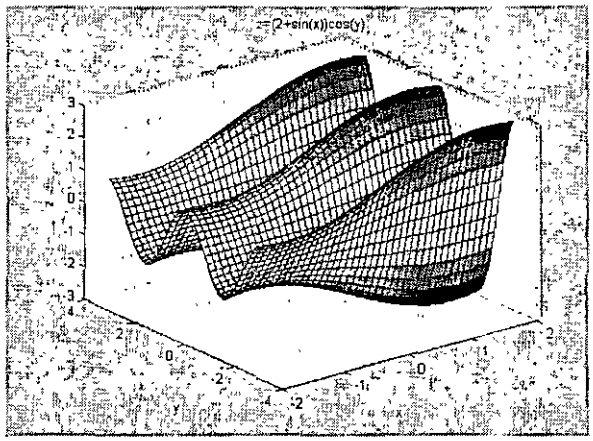
La orden for, cuya sintaxis es:

```
for índice = valor inicial:incremento:valor final
        conjunto de instrucciones
end
```

permite repetir un conjunto de órdenes en función del valor inicial, el valor final y un incremento de la variable índice. El incremento es opcional, si es omitido, se considera igual a 1.
Una vez generada la matriz , se emplea una orden para graficar la superficie, por ejemplo, surf:

```
surf(x,y,z ')
```

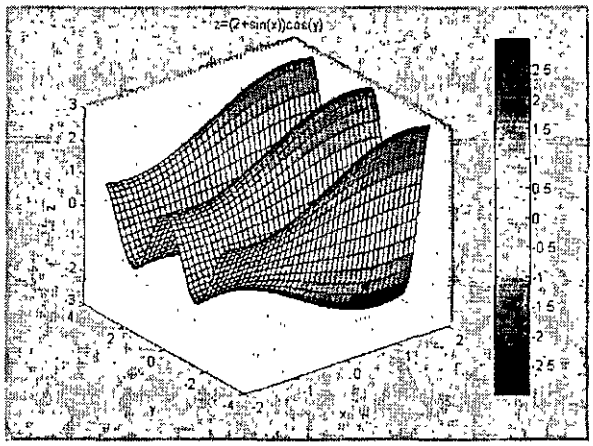
Cuando los dos primeros argumentos son vectores, el tercero debe ser la transpuesta de la matriz con los valores de la variable dependiente. Opcionalmente, se pueden indicar etiquetas para los ejes y un título para la gráfica.
La gráfica obtenida mediante las órdenes anteriores es:



Colorbar despliega en la ventana actual una escala que relaciona los colores con los valores de la variable dependiente. La última figura después de ejecutar

colorbar

queda de la siguiente forma



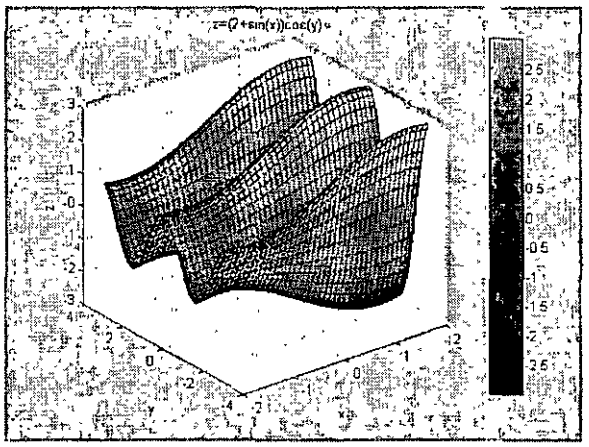
Con colormap es posible especificar un mapa de colores. Los mapas de colores disponibles son:

hsv	hot	gray	bone	copper	pink
white	flag	lines	colorcube	vga	jet
prism	cool	autumn	spring	winter	summer

Al ejecutar la orden

colormap(winter)

la última figura ahora se ve así:



7.- Solución de ecuaciones no lineales

7.1.- Solución numérica de ecuaciones de una variable f(x)=0

Las siguientes funciones son utilizadas para encontrar raíces o ceros de ecuaciones del tipo f(x)=0

fzero	Encuentra una raíz de una función
edit	Inicia la ejecución del editor de texto de MATLAB
type archivo	Despliega el contenido de un archivo de texto
inline	Construye una función en línea a partir de una expresión contenida en una cadena de caracteres

Un archivo diary que muestra el uso de fzero es:

```
% Este es un archivo diary
%
help fzero

 FZERO  Scalar nonlinear zero finding.
   X = FZERO(FUN,X0) tries to find a zero of the function FUN near X0.
   FUN accepts real scalar input X and returns a real scalar function value F
   evaluated at X.  The value X returned by FZERO is near a point where FUN
   changes sign (if FUN is continuous), or NaN if the search fails.

   X = FZERO(FUN,X0), where X is a vector of length 2, assumes X0 is an
   interval where the sign of FUN(X0(1)) differs from the sign of FUN(X0(2)).
   An error occurs if this is not true.  Calling FZERO with an interval
   guarantees FZERO will return a value near a point where FUN changes
   sign.

   X = FZERO(FUN,X0), where X0 is a scalar value, uses X0 as a starting guess.
   FZERO looks for an interval containing a sign change for FUN and
   containing X0.  If no such interval is found, NaN is returned.
   In this case, the search terminates when the search interval
   is expanded until an Inf, NaN, or complex value is found.

   X = FZERO(FUN,X0,OPTIONS) minimizes with the default optimization
   parameters replaced by values in the structure OPTIONS, an argument
   created with the OPTIMSET function.  See OPTIMSET for details.  Used
```

options are Display and TolX. Use OPTIONS = [] as a place holder if no options are set.

X = FZERO(FUN,X0,OPTIONS,P1,P2,...) allows for additional arguments which are passed to the function, F=feval(FUN,X,P1,P2,...). Pass an empty matrix for OPTIONS to use the default values.

[X,FVAL]= FZERO(FUN,...) returns the value of the objective function, described in FUN, at X.

[X,FVAL,EXITFLAG] = FZERO(...) returns a string EXITFLAG that describes the exit condition of FZERO.
If EXITFLAG is:
  > 0 then FZERO found a zero X
  < 0 then no interval was found with a sign change, or
     NaN or Inf function value was encountered during search
     for an interval containing a sign change, or
     a complex function value was encountered during search
     for an interval containing a sign change.

[X,FVAL,EXITFLAG,OUTPUT] = FZERO(...) returns a structure OUTPUT with the number of iterations taken in OUTPUT.iterations.

Examples
  FUN can be specified using @:
    X = fzero(@sin,3)
  returns pi.
    X = fzero(@sin, 3, optimset('disp','iter'))
  returns pi, uses the default tolerance and displays iteration information.

  FUN can also be an inline object:
    X = fzero(inline('sin(3*x)'),2);

Limitations
    X = fzero(inline('abs(x)+1'), 1)
  returns NaN since this function does not change sign anywhere on the
  real axis (and does not have a zero as well).
    X = fzero(@tan,2)
  returns X near 1.5708 because the discontinuity of this function near the
  point X gives the appearance (numerically) that the function changes sign at X.

See also ROOTS, FMINBND, @, INLINE.

```
f='x^2-5*cos(2*x)';          % definicion de una expresion
```

```
fzero(f,[0,3])  % calculo de una raiz de f contenida en el intervalo [0,3]

ans =

   0.7317

fzero(f,[.5,1])  % calculo de una raiz de f contenida en el intervalo [.5,1]

ans =

   0.7317

help inline

 INLINE Construct INLINE object.
    INLINE(EXPR) constructs an inline function object from the
    MATLAB expression contained in the string EXPR.  The input
    arguments are automatically determined by searching EXPR
    for variable names (see SYMVAR). If no variable exists, 'x'
    is used.

    INLINE(EXPR, ARG1, ARG2, ...) constructs an inline
    function whose input arguments are specified by the
    strings ARG1, ARG2, ...  Multicharacter symbol names may
    be used.

    INLINE(EXPR, N), where N is a scalar, constructs an
    inline function whose input arguments are 'x', 'P1',
    'P2', ..., 'PN'.

    Examples:
      g = inline('t^2')
      g = inline('sin(2*pi*f + theta)')
      g = inline('sin(2*pi*f + theta)', 'f', 'theta')
      g = inline('x^P1', 1)

    See also SYMVAR.


% x es la raiz encontrada
% fx es el valor de f evaluada en la raiz x
% se provee de una estimación inicial para la raiz
[x,fx]=fzero(inline(f),.7)
```

```
x =

   0.7317


fx =

  6.6613e-016

% calculo de las raices de la funcion
g(x)=(5x-6.4)/((x-1.3)^2+.002)+9x/(x^3+.03)-(x-0.4)/((x-.92)^2+.005)=0

edit g              % edicion de una funcion g

type g.m            % se muestra el contenido del archivo g.m

function y=g(x)
%
%  Esta funcion sera resuelta con fzero
%
y=(5*x-6.4)/((x-1.3)^2+.002)+9*x/(x^3+.03)-(x-.4)/...
    ((x-.92)^2+0.005);

% una funcion debe comenzar con el encabezado
% function resultado = nombre de la funcion( lista de argumentos)

% con la ayuda de una grafica se pueden proponer valores iniciales adecuados
% para el calculo de raices con fzero

% Un @ crea un manejador (handle) para una función lo cual permite pasar como
% argumento el nombre de una función

fplot(@g,[0,3])
grid

fzero(@g,.2)

ans =

   0.0112

fzero(@g,.7)

ans =
```

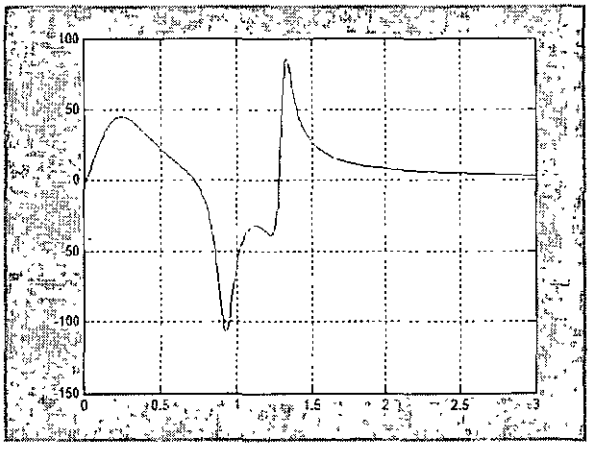
0.7248

fzero(@g,1.2)

ans =

1.2805

La gráfica obtenida arriba es



7.2.- Solución numérica de sistemas de ecuaciones no lineales

La función fsolve resuelve un sistema de ecuaciones no lineales. Esta función es parte de la caja de herramientas Optimization (Optimization Toolbox)

fsolve	Solución numérica de un sistema de ecuaciones no lineales con el método de los mínimos cuadrados

Un archivo diary que ilustra el uso de fsolve es:

% Este es un archivo diary
%

help fsolve

FSOLVE Solves nonlinear equations by a least squares method.

FSOLVE solves equations of the form:

F(X)=0 where F and X may be vectors or matrices.

X=FSOLVE(FUN,X0) starts at the matrix X0 and tries to solve the equations in FUN. FUN accepts input X and returns a vector (matrix) of equation values F evaluated at X.

X=FSOLVE(FUN,X0,OPTIONS) minimizes with the default optimization parameters replaced by values in the structure OPTIONS, an argument created with the OPTIMSET function. See OPTIMSET for details. Used options are Display, TolX, TolFun, DerivativeCheck, Diagnostics, Jacobian, JacobMult, JacobPattern, LineSearchType, LevenbergMarquardt, MaxFunEvals, MaxIter, DiffMinChange and DiffMaxChange, LargeScale, MaxPCGIter, PrecondBandWidth, TolPCG, TypicalX. Use the Jacobian option to specify that FUN also returns a second output argument J that is the Jacobian matrix at the point X. If FUN returns a vector F of m components when X has length n, then J is an m-by-n matrix where J(i,j) is the partial derivative of F(i) with respect to x(j). (Note that the Jacobian J is the transpose of the gradient of F.)

X=FSOLVE(FUN,X0,OPTIONS,P1,P2,...) passes the problem-dependent parameters P1,P2,... directly to the function FUN: FUN(X,P1,P2,...). Pass an empty matrix for OPTIONS to use the default values.

[X,FVAL]=FSOLVE(FUN,X0,...) returns the value of the objective function at X.

[X,FVAL,EXITFLAG]=FSOLVE(FUN,X0,...) returns a string EXITFLAG that describes the exit condition of FSOLVE.
If EXITFLAG is:
> 0 then FSOLVE converged to a solution X.
0 then the maximum number of function evaluations was reached.
< 0 then FSOLVE did not converge to a solution.

[X,FVAL,EXITFLAG,OUTPUT]=FSOLVE(FUN,X0,...) returns a structure OUTPUT with the number of iterations taken in OUTPUT.iterations, the number of function evaluations in OUTPUT.funcCount, the algorithm used in OUTPUT.algorithm, the number of CG iterations (if used) in OUTPUT.cgiterations, and the first-order optimality (if used) in OUTPUT.firstorderopt.

[X,FVAL,EXITFLAG,OUTPUT,JACOB]=FSOLVE(FUN,X0,...) returns the Jacobian of FUN at X.

```
Examples
  FUN can be specified using @:
    x = fsolve(@myfun,[2 3 4],optimset('Display','iter'))

where MYFUN is a MATLAB function such as:

    function F = myfun(x)
    F = sin(x);

FUN can also be an inline object:

    fun = inline('sin(3*x)');
    x = fsolve(fun,[1 4],optimset('Display','off'));

See also OPTIMSET, LSQNONLIN, @, INLINE.
```

```
% solucion del sistema
% x1^2+x2^2-4 = 0
% x1*x2-1=0

% ecuaciones.m es una funcion que describe al sistema de ecuaciones
% el resultado es devuelto como un vector columna

edit ecuaciones

type ecuaciones.m

function F=ecuaciones(x)
%
%  Sistema de ecuaciones no lineales
%
F=[x(1)^2+x(2)^2-4;x(1)*x(2)-1];

fsolve(@ecuaciones,[2,0],optimset('fsolve'))
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.

ans =

   1.9319    0.5176

% x es un vector que contiene la solucion encontrada
% fx es un vector con los valores de las ecuaciones evaluadas en la raiz x
```

```
[x,fx]=fsolve(@ecuaciones,[2;0],optimset('fsolve'))
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.

x =

   1.9319
   0.5176


fx =

  1.0e-011 *

   0.2307
  -0.1125

% status=1 implica que el calculo de las raices fue exitoso
[x,fx,status]=fsolve(@ecuaciones,[2;0],optimset('fsolve'))
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.

x =

   1.9319
   0.5176


fx =

  1.0e-011 *

   0.2307
  -0.1125


status =

     1

% en salida se devuelven algunas estadísticas finales
[x,fx,status,salida]=fsolve(@ecuaciones,[2;0],optimset('fsolve'))
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
```

```
x =

   1.9319
   0.5176


fx =

  1.0e-011 *

   0.2307
  -0.1125


status =

     1


salida =

       iterations: 5
        funcCount: 15
        algorithm: 'trust-region dogleg'
    firstorderopt: 8.3294e-012

% se especifican las opciones que serán empleadas por fsolve
opciones=optimset('Display','iter','LevenbergMarquardt','on');

fsolve(@ecuaciones,[2;0],opciones)

                                        Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)          step         optimality   radius
     1          3              1                          2             1
     2          6         0.0625             0.5             1             1
     3          9      2.3534e-005       0.0687184        0.0177          1.25
     4         12      1.18853e-011      0.00177026       14e-005         1.25
     5         15      6.587e-024        1.50231e-006     8.33e-012       1.25
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.

ans =

   1.9319
```

8.- Integración numérica

Las siguientes funciones son utilizadas para aproximar el valor de integrales definidas:

trapz	Integración numérica con la regla de los trapecios
quad	Integración numérica con cuadratura adaptativa de Simpson
quadl	Integración numérica con cuadratura adaptativa de Lobatto
dblquad	Evaluación numérica de una integral doble
triplequad	Evaluación numérica de una integral triple

Un archivo diary que muestra el uso de estas funciones es:

```
% Este es un archivo diary
%

help trapz

 TRAPZ  Trapezoidal numerical integration.
   Z = TRAPZ(Y) computes an approximation of the integral of Y via
   the trapezoidal method (with unit spacing).  To compute the integral
   for spacing different from one, multiply Z by the spacing increment.

   For vectors, TRAPZ(Y) is the integral of Y. For matrices, TRAPZ(Y)
   is a row vector with the integral over each column. For N-D
   arrays, TRAPZ(Y) works across the first non-singleton dimension.

   Z = TRAPZ(X,Y) computes the integral of Y with respect to X using
   the trapezoidal method.  X and Y must be vectors of the same
   length, or X must be a column vector and Y an array whose first
   non-singleton dimension is length(X).  TRAPZ operates along this
   dimension.

   Z = TRAPZ(X,Y,DIM) or TRAPZ(Y,DIM) integrates across dimension DIM
   of Y. The length of X must be the same as size(Y,DIM)).

   Example: If Y = [0 1 2
                    3 4 5]

   then trapz(Y,1) is [1.5 2.5 3.5] and trapz(Y,2) is [2
```

8];

See also SUM, CUMSUM, CUMTRAPZ.

```
x=0:.1:2;
y=x.^2;

% el resultado de trapz debe ser multiplicado por el espaciamiento entre los valores x
area=trapz(y)*(x(2)-x(1))

area =

   2.6700

help quad

 QUAD  Numerically evaluate integral, adaptive Simpson quadrature.
   Q = QUAD(FUN,A,B) tries to approximate the integral of function
   FUN from A to B to within an error of 1.e-6 using recursive
   adaptive Simpson quadrature.  The function Y = FUN(X) should
   accept a vector argument X and return a vector result Y, the
   integrand evaluated at each element of X.

   Q = QUAD(FUN,A,B,TOL) uses an absolute error tolerance of TOL
   instead of the default, which is 1.e-6.  Larger values of TOL
   result in fewer function evaluations and faster computation,
   but less accurate results.  The QUAD function in MATLAB 5.3 used
   a less reliable algorithm and a default tolerance of 1.e-3.

   [Q,FCNT] = QUAD(...) returns the number of function evaluations.

   QUAD(FUN,A,B,TOL,TRACE) with non-zero TRACE shows the values
   of [fcnt a b-a Q] during the recursion.

   QUAD(FUN,A,B,TOL,TRACE,P1,P2,...) provides for additional
   arguments P1, P2, ... to be passed directly to function FUN,
   FUN(X,P1,P2,...).  Pass empty matrices for TOL or TRACE to
   use the default values.

   Use array operators .*, ./ and .^ in the definition of FUN
   so that it can be evaluated with a vector argument.

   Function QUADL may be more efficient with high accuracies
   and smooth integrands.
```

```
  Example:
    FUN can be specified as:

    An inline object:
      F = inline('1./(x.^3-2*x-5)');
      Q = quad(F,0,2);

    A function handle:
      Q = quad(@myfun,0,2);
      where myfun.m is an M-file:
        function y = myfun(x)
        y = 1./(x.^3-2*x-5);

  See also QUADL, DBLQUAD, TRIPLEQUAD, INLINE, @.

edit fintegra     % fintegra.m define a la funcion por integrar

type fintegra

function f=fintegra(x)
%
%  f=fintegra(x)
%  Esta es una funcion que sera utilizada como integrando en quad y quadl
%
f=x.^2.*cos(3*x);

area=quad(@fintegra,0,pi)    % cuadratura adaptativa de Simpson

area =

  -0.6981

area=quadl(@fintegra,0,pi)    % cuadratura adaptativa de Lobatto

area =

  -0.6981

area=quad(inline('x.^2'),0,2)

area =

   2.6667
```

help dblquad

DBLQUAD Numerically evaluate double integral.
DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX) evaluates the double integral of FUN(X,Y) over the rectangle XMIN <= X <= XMAX, YMIN <= Y <= YMAX. FUN(X,Y) should accept a vector X and a scalar Y and return a vector of values of the integrand.

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL) uses a tolerance TOL instead of the default, which is 1.e-6.

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL,@QUADL) uses quadrature function QUADL instead of the default QUAD.
DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL,@MYQUADF) uses your own quadrature function MYQUADF instead of QUAD. MYQUADF should have the same calling sequence as QUAD and QUADL.

DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,TOL,@QUADL,P1,P2,...) passes the extra parameters to FUN(X,Y,P1,P2,...).
DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,[],[],P1,P2,...) is the same as DBLQUAD(FUN,XMIN,XMAX,YMIN,YMAX,1.e-6,@QUAD,P1,P2,...)

Example:
FUN can be an inline object or a function handle.

```
Q = dblquad(inline('y*sin(x)+x*cos(y)'), pi, 2*pi, 0, pi)
```

or

```
Q = dblquad(@integrnd, pi, 2*pi, 0, pi)
```

where integrnd.m is an M-file:

```
function z = integrnd(x, y)
z = y*sin(x)+x*cos(y);
```

This integrates y*sin(x)+x*cos(y) over the square pi <= x <= 2*pi, 0 <= y <= pi. Note that the integrand can be evaluated with a vector x and a scalar y .

Nonsquare regions can be handled by setting the integrand to zero outside of the region. The volume of a hemisphere is

```
dblquad(inline('sqrt(max(1-(x.^2+y.^2),0))'),-1,1,-1,1)
```

or

```
dblquad(inline('sqrt(1-(x.^2+y.^2)).*(x.^2+y.^2<=1)'),-1,1,-1,1)
```

See also QUAD, QUADL, TRIPLEQUAD, INLINE, @.

```
% calculo del area de un circulo de radio unitario
dblquad(inline('r*theta^0'),0,1,0.2*pi)

ans =

   3.1416
```