Laboratorio 9

Matlab

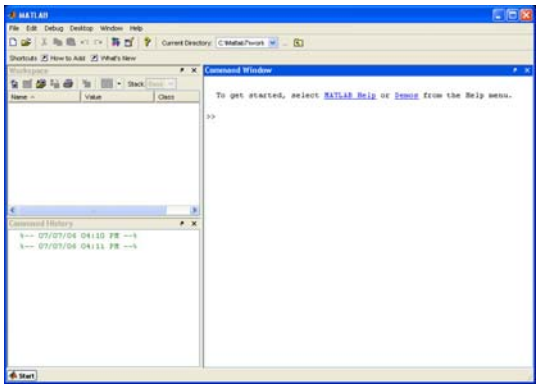
MATLAB es el nombre abreviado de "Matriz LABoratory". MATLAB es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular, puede también trabajar con números escalares, tanto reales como complejos. Una de las capacidades más atractivas es la de realizar un amplia variedad de gráficos en dos y tres dimensiones. MATLAB tiene también un lenguaje de programación propio.

- Comandos Iniciales
- Operaciones con Matrices y Vectores
    - Definición de matrices y vectores
    - Operaciones básicas
    - Solución de sistemas de ecuaciones lineales simultáneas
    - Números complejos
    - Tipos predefinidos de matrices
    - Creación de una matriz a partir de otras
    - Operador dos puntos
    - Operadores relacionales
    - Operadores lógicos
    - Funciones de Librería
    - Cadenas de Caracteres
    - Estructuras
    - Vectores de Celdas
- Programación en MATLAB

Actividades

Semana 1

1. Cree una carpeta en el escritorio con el nombre de Laboratorio 9
2. Ejecute el programa MATLAB



3. Ajuste el directorio de trabajo (Current Directory) de tal manera que señale a la carpeta que creó en el escritorio

Operaciones Básicas con Matrices

4. Ejecute el comando »A=[1 2 3; 4 5 6; 7 8 9]. Con él es estará creando una matriz de 3 filas por 3 columnas con los valores indicados.

```
>> A=[1 2 3;4 5 6;7 8 9]

A =

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

5. Para obtener la matriz transpuesta, ejecute el comando »A'. Como el resultado no se ha asignado a ninguna variable, MATLAB responde con ans

```
>> A'

ans =

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

6. También puede calcular la matriz transpuesta y asignarla a otra variable; ejecute el comando »B=A'
7. Para multiplicar dos matrices, ejecute un comando como el siguiente »B*A

```
>> B*A

ans =

    66    78    90
    78    93   108
    90   108   126
```

8. Para acceder a un elemento específico dentro de la matriz, escriba el comando `»A(1,2)`. El primer número indica la fila y el segundo la columna.
9. Cree una matriz llamada A con los siguientes valores {1 4 -3; 2 1 5; -2 5 3}
10. Para calcular la inversa de la matriz A y asignarla a una matriz B, ejecute el comando `»B=inv(A)`

```
>> B=inv(A)

B =

    0.1803    0.2213   -0.1885
    0.1311    0.0246    0.0902
   -0.0984    0.1066    0.0574
```

11. Calcule el producto de A por B; siendo que B es la inversa de A, el resultado del producto debe ser una matriz identidad
12. Para definir un vector fila, ejecute el comando `»x=[10 20 30]`
13. Para definir un vector columna, ejecute el comando `»y=[11;12;13]`

Solución de Sistemas de Ecuaciones Lineales Simultáneas

14. Para definir un sistema de ecuaciones simultáneas como el que se muestra, ingrese el siguiente comando `»A=[1 1 1;2 -1 1;-1 1 -1], b=[3;5;7]` donde A corresponde a la matriz de los coeficientes de las variables y b a los términos independientes

$$\begin{aligned} x + y + z &= 3 \\ 2x - y + z &= 5 \\ -x + y - z &= 7 \end{aligned}$$

15. Para obtener la solución del sistema de ecuaciones, ejecute el comando `»x=A\b`. Alternativamente puede utilizar el comando `»x=inv(A)*b`

```
>> x=A\b

x =

   12.0000
    5.0000
  -14.0000
```

Números Complejos

16. Ejecute el comando `»sqrt(-4)`. La raíz cuadrada de un número negativo tiene solución en el dominio de los números complejos; es por eso la solución 0+2i que arroja MATLAB
17. Ejecute el comando `»a=3+4i`
18. Ejecute el comando `»b=1-i`
19. Ejecute el comando `»a*b`

```
>> a*b

ans =

   7.0000 + 1.0000i
```

20. También puede crear un número complejo utilizando el comando `»complex(1,2)`
21. Puede crear matrices con números complejos utilizando un comando como el siguiente:
    `»A=[1+2i 2+3i;-1+i 2-3i]`

Tipos Predefinidos de Matrices

22. Ejecute los siguientes comandos:

- `»eye(4)` genera una matriz identidad de tamaño 4x4
- `»zeros(3,5)` genera una matriz nula de 3x5
- `»zeros(4)` genera una matriz nula de 4x4
- `»ones(4)` genera una matriz de unos de 3x3
- `»ones(2,4)` genera una matriz de unos de 2x4
- `»linspace(1,5,4)` genera un vector de 4 valores equiespaciados entre 1 y 5
- `»logspace(1,5,4)` genera un vector de 4 valores equiespaciados logaritmicamente entre 10 ^ 1 y 10 ^ 5
- `»rand(3)` genera una matriz de 3x3 con números aleatorios entre 0 y 1
- `»rand(2,5)` genera una matriz de 2x5 con números aleatorios entre 0 y 1
- `»randn(4)` genera una matriz de 4x4 con números aleatorios de acuerdo con una distribución normal con valor medio 0 y varianza 1
- `»magic(4)` genera una matriz de 4x4 con los números desde 1 hasta 4x4 de tal manera que todas las columnas y filas suman el mismo valor
- `»hilb(5)` genera una matriz de Hilbert de 5x5. En esta matriz, cada elemento es calculado con base a la expresión 1/(i+j-1)
- `»invhilb(5)` genera una matriz inversa de Hilbert de 5x5

- `»x=[1 2 3],y=[4 5 6]`
- `»kron(x,y)` genera una matriz con todos los productos de los elementos del vector x por los elementos del vector y. Equivalente a x'*y donde x e y son vectores fila
- `»pol=[1 5 7]`
- `»compan(pol)` construye una matriz cuyo polinomio característica tiene como coeficientes los elementos del vector pol (ordenados de mayor a menor grado)
- `»v=[4 5 6]`
- `»vander(v)` construye la matriz de Vandermonde a partir del vector v (las columnas son las potencias de los elementos de dicho vector)

Formación de una Matriz a partir de Otras

23. Ejecute los siguientes comandos:

- `»A=[1 2 3;4 5 6;7 8 9]`
- `»[m,n]=size(A)` devuelve el número de filas y de columnas de la matriz A
- `»x=[1 5 9]`
- `»n= length(x)` calcula el número de elementos en el vector x
- `»zeros(size(A))` construye una matriz de ceros del mismo tamaño que la matriz A
- `»ones(size(A))` construye una matriz de unos del mismo tamaño que la matriz A
- `»A=diag(x)` crea una matriz diagonal A cuyos elementos (de la diagonal) son los existentes en el vector x
- `»x=diag(A)` crea un vector x a partir de los elementos de la diagonal de la matriz A
- `»diag(diag(A))` crea una matriz diagonal a partir de la diagonal de la matriz A
- `B=[9 8 7;6 5 4;3 2 1]`
- `»blkdiag(A,B)` crea una matriz diagonal de submatrices a partir de las matrices A y B
- `»A=B`
- `»triu(A)` crea una matriz triangular superior a partir de una matriz A
- `»tril(A)` crea una matriz triangular inferior a partir de una matriz A
- `»rot90(A,k)` gira k*90 grados la matriz rectangular A en sentido antihorario. k es un número entero que puede ser negativo. Si se omite, se supone k=1. Pruebe el comando con diversos valores para k
- `»flipud(A)` halla la matriz simétrica de A con respecto al eje horizontal
- `»fliplr(A)` halla la matriz simétrica de A con respecto al eje vertical
- `»reshape(A,m,n)` Cambia el tamaño de la matriz A, devolviendo una matriz de tamaño mxn; el número total de elementos no puede cambiar. Pruebe el comando con diferentes valores de m y n

Direccionamiento de Vectores y Matrices a partir de Vectores

24. Ejecute el comando `»v=[1 3 4];x=rand(1,6)`
25. Ejecute el comando `»x(v)`. Explique el resultado
26. Ejecute el comando `»f=[2 4];c=[1 2];A=magic(4)`
27. Ejecute el comando `»A(f,c)`. Explique el resultado

Operador dos Puntos (:)

28. Ejecute el comando `»x=1:10`
29. Ejecute el comando `»x=1:2:10`
30. Ejecute el comando `»x=1:1.5:10`
31. Ejecute el comando `»x=10:-1:1`. Explique el resultado
32. Ejecute los comandos `»x=[0.0:pi/50:2*pi]';y=sin(x);z=cos(x);[x y z]` Explique el resultado
33. Ejecute los comandos `»A=magic(6);A(6,1:4)`. Explique el resultado

Operadores Relacionales

En MATLAB se dispone de los siguientes operadores relacionales

- < menor que
- > mayor que
- <= menor o igual que
- >= mayor o igual que
- == igual que
- ~= distinto de

34. Ejecute el comando `»A=[1 2;0 3];B=[4 2;1 5]`
35. Ejecute el comando `»A==B`. Explique el resultado
36. Ejecute el comando `»A~=B`. Explique el resultado

Operadores Lógicos

En MATLAB se dispone de los siguientes operadores lógicos

- & operador y (and)
- > operador o (or)
- <= negación lógica

37. Ejecute el comando `»A=[1 2;0 3`
38. Ejecute el comando `~A`. Explique el resultado

Funciones de Librería

En MATLAB se dispone de las siguientes funciones de librería

- Funciones matemáticas elementales
- Funciones especiales
- Funciones matriciales elementales
- Funciones matriciales específicas
- Funciones para la descomposición y/o factorización de matrices
- Funciones para análisis estadístico de datos

- Funciones para el análisis de polinomios
- Funciones para integración de ecuaciones diferenciales ordinarias
- Resolución de ecuaciones no-lineales y optimización
- Integración numérica
- Funciones para procesamiento de señal

39. Ejecute el comando `»x=[-pi:pi/10:pi]'`
40. Ejecute el comando `»y=sin(x)`. Explique el resultado

```
>> y=sin(x)

y =

   -0.0000
   -0.3090
   -0.5878
   -0.8090
   -0.9511
   -1.0000
   -0.9511
   -0.8090
   -0.5878
   -0.3090
         0
    0.3090
    0.5878
    0.8090
    0.9511
    1.0000
    0.9511
    0.8090
    0.5878
    0.3090
    0.0000
```

41. Ejecute el comando `»sum(x)`. Explique el resultado
42. Ejecute el comando `»max(x)`. Explique el resultado
43. Ejecute el comando `»min(x)`. Explique el resultado
44. Ejecute el comando `»std(x)`. Explique el resultado
45. Ejecute el comando `»A=magic(4);trace(A)`. Explique el resultado
46. Ejecute el comando `»[L,U]=lu(A)`. Explique el resultado

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

L =

    1.0000         0         0         0
    0.3125    0.7685    1.0000         0
    0.5625    0.4352    1.0000    1.0000
    0.2500    1.0000         0         0


U =

   16.0000    2.0000    3.0000   13.0000
         0   13.5000   14.2500   -2.2500
         0         0   -1.8889    5.6667
         0         0         0    0.0000
```

Cadena de Caracteres

47. Puede asignar una cadena de caracteres a una variable, utilizando el siguiente comando
    `»s='cadena de caracteres'`
48. Ejecute el comando `»size(s)`. Explique el resultado
49. Ejecute el comando `»double(s)`. Explique el resultado
50. Ejecute el comando `»findstr(s,'de')`. Explique el resultado
51. Ejecute el comando `»c='cadena'`
52. Ejecute el comando `»strcmp(s,c)`. Explique el resultado

Estructura

Una estructura es una agrupación de datos de tipos diferentes bajo un mismo nombre. Los datos se llaman miembros o campos. Una estructura es un nuevo tipo de dato, del que luego se pueden instanciar variables.

53. Ejecute el comando `»alu.nombre='Pedro'`
54. Ejecute el comando `»alu.carnet=75482`
55. Ejecute el comando
    `»alum(10)=struct('nombre','Pedro','carnet',76589)`
56. Ejecute el comando `»alum(5).nombre= 'José';alum(5).carnet=77524`
57. Ejecute el comando `»alum(5)`. Explique el resultado

Vectores o Matrices de Celdas

Un vector de celdas es aquel cuyos elementos son a su vez variables de cualquier tipo. En un arreglo ordinario, todos los elementos son del mismo tipo. Sin embargo, en un arreglo de celdas, el primer elemento puede ser un número, el segundo una matriz y el tercero una cadena de caracteres y así sucesivamente.

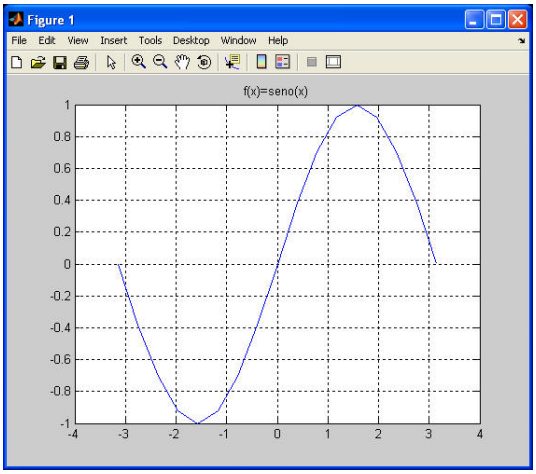
58. Ejecute el comando `»Z={pi,'Informática',rand(3,3)}`.
59. Ejecute el comando `»celldisp(Z)`
60. Ejecute el comando `»Z{1}`. Explique el resultado
61. Ejecute el comando `»Z{1}(2,2)`. Explique el resultado

Semana 2

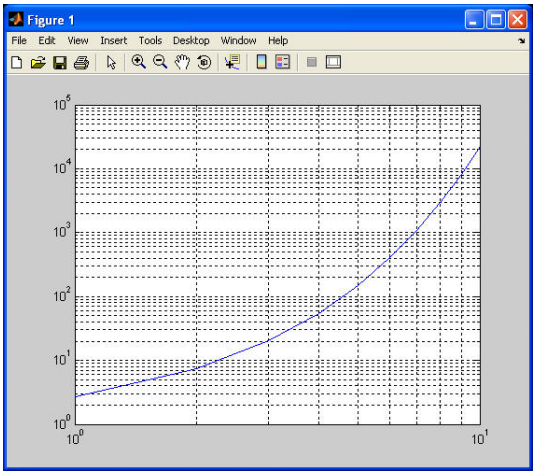
62. Ejecute el programa MATLAB
63. Ajuste el directorio de trabajo (Current Directory) de tal manera que señale a la carpeta que creó en el escritorio

Gráficos Bidimensionales

64. Ejecute los comandos `»x=[-pi:pi/8:pi]';y=sin(x);[x y]`
65. Ejecute los comandos `»plot(x,y);grid;title('f(x)=seno(x)')`



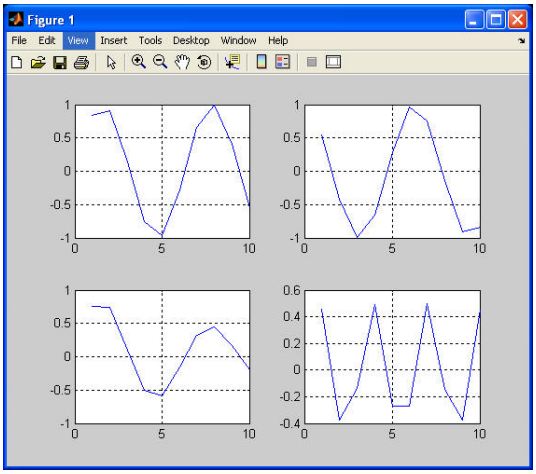
66. Ejecute los comandos `»x=[1:10]';y=exp(x);[x y]`
67. Ejecute los comandos `»loglog(x,y);grid;title('f(x)=exp(x)')`



68. Ejecute el comando `»plot(eig(rand(20,20)),'+');gris`
69. Ejecute los siguientes comandos

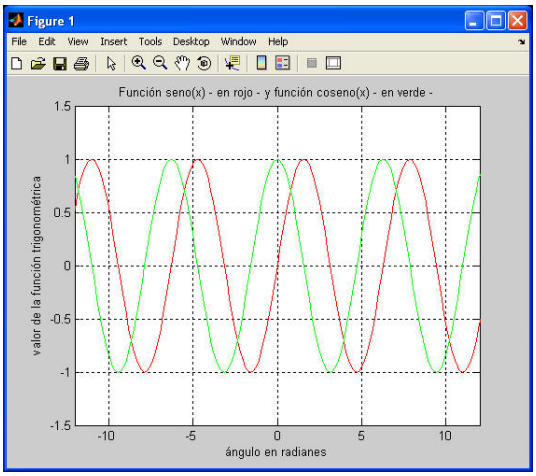
- `»y=sin(x);z=cos(x);w=exp(-x*.1).*y;v=y.*z`
- `»subplot(2,2,1),plot(x,y),gris`
- `»subplot(2,2,2),plot(x,z),grid`

- `»subplot(2,2,3),plot(x,w),grid`
- `»subplot(2,2,4),plot(x,v),grid`



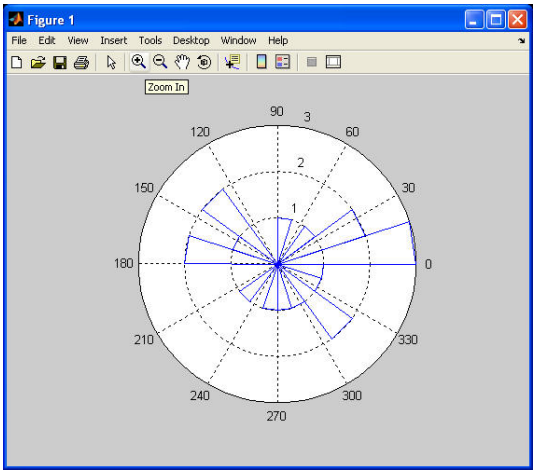
70. Ejecute los siguientes comandos

- `»x=[-4*pi:pi/20:4*pi]`
- `»plot(x,sin(x),'r',x,cos(x),'g')`
- `»plot(x,sin(x),'r',x,cos(x),'g')`
- `»title('Función seno(x) - en rojo - y función coseno(x) - en verde -')`
- `»xlabel('ángulo en radianes'),figure(gcf)`
- `»ylabel('valor de la función trigonométrica'),figure(gcf)`
- `»axis[-12,12,-1.5,1.5]),figure(gcf)`. Explique el resultado
- `»axis('normal'),figure(gcf)`. Explique el resultado
- `»axis('square'),figure(gcf)`. Explique el resultado
- `»axis('off'),figure(gcf)`. Explique el resultado
- `»axis('on'),figure(gcf)`. Explique el resultado
- `»axis('on'),grid,figure(gcf)`. Explique el resultado

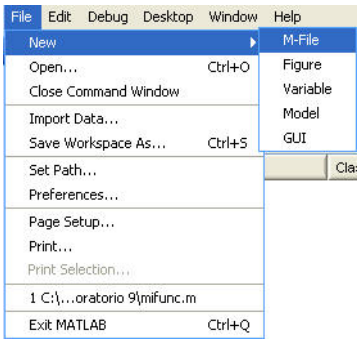


71. Ejecute los siguientes comandos

- `»x=[rand(1,100)*10]`
- `»plot(x)`
- `»bar(x)`. Explique el resultado
- `»stairs(x)`. Explique el resultado
- `»hist(x)`. Explique el resultado
- `»hist(x,20)`. Explique el resultado
- `»alfa=(rand(1,20)-0.5)*2*pi`
- `»rose(alfa)`. Explique el resultado



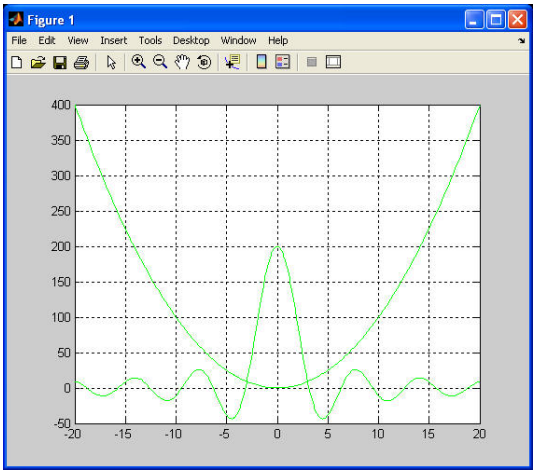
72. Seleccione «M-File» en la alternativa «New» del menú «File». Esto permitirá crear un archivo .m para definir una función



73. En el editor del archivo .m, escriba los siguientes comandos

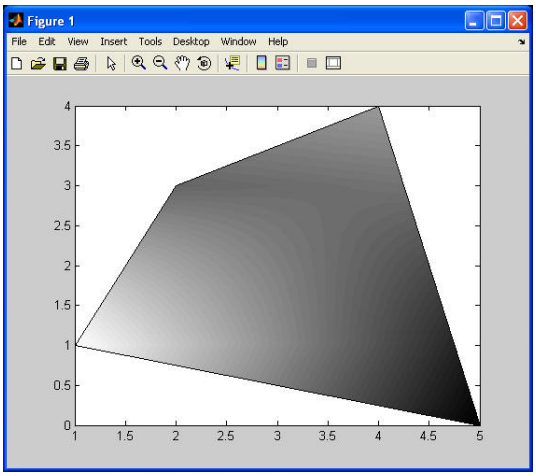
- `function y=mifunc(x)`
- `y(:,1)=200*sin(x)./x;`
- `y(:,2)=x.^2;`

74. Guarde el archivo con el nombre mifunc.m
75. Ejecute el comando `»fplot('mifunc(x)',[-20 20],'g')`



76. Ejecute los siguientes comandos

- `»x=[1 5 4 2];y=[1 0 4 3]`
- `»fill(x,y,'r')`
- `»colormap(gray),fill(x,y,[1 0.5 0.8 0.7])`. Explique el resultado



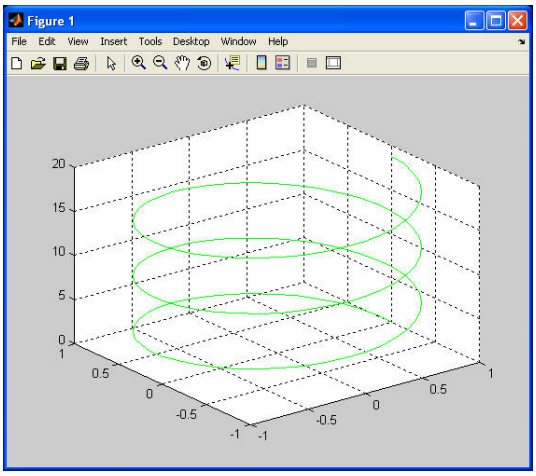
77. Ejecute los siguientes comandos

- `»M=moviein(17)`
- `»x=[-2*pi:0.1:2*pi]'`
-
```
»for j=1:17
y=sin(x+j*pi/8);
plot(x,y);
M(:,j)=getframe;
end
```
- `»movie(M,10,15)`. Explique el resultado

Gráficos Tridimensionales

78. Ejecute los siguientes comandos:

- `»fi=[0:pi/20:6*pi];`
- `»plot3(cos(fi),sin(fi),fi,'g');gris`



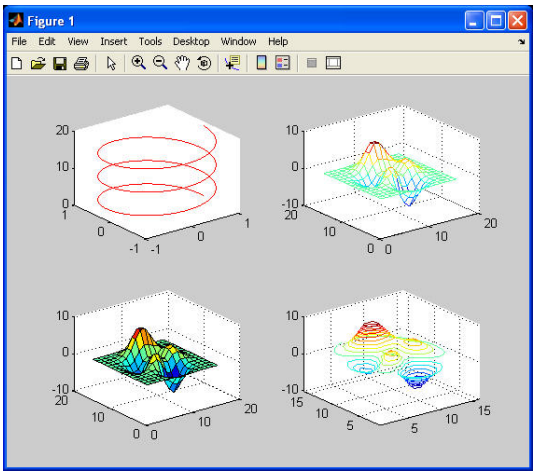
79. Cree un archivo .m con el nombre prueba3d.m que contenga los siguientes comandos

- `function z=prueba3d(x,y)`
- `z=3*(1-x).^2.*exp(-(x.^2)-(y+1).^2)-10*(x/5-x.^3-y.^5).*exp(-x.^2-y.^2)-1/3*exp(-(x+1).^2-y.^2);`

80. Cree un archivo .m con el nombre prueba3dFC.m que contenga los siguientes comandos

- `x=[-3:0.4:3];y=x;`
- `close`
- `subplot(2,2,1)`
- `figure(gcf),fi=[0:pi/20:6*pi];`
- `plot3(cos(fi),sin(fi),fi,'r')`
- `[X,Y]=meshgrid(x,y)`
- `Z=prueba3d(X,Y)`
- `subplot(2,2,2)`
- `figure(gcf),mesh(Z)`
- `subplot(2,2,3)`
- `figure(gcf),surf(Z)`
- `subplot(2,2,4)`
- `figure(gcf),contour3(Z,16)`

81. Ejecute el comando `»prueba3dFC`



Funciones para Cálculos con Polinomios

82. Para ingresar el polinomio $X^4 - 8X^2 + 6X - 10 = 0$ ejecute el comando `»pol=[1 0 -8 6 -10]`
83. Para evaluar el polinomio para X=1, ejecute el comando `»polyval(pol,1)`
84. Para calcular las raíces del polinomio, ejecute el comando `»roots(pol)`

```
>> roots(pol)

ans =

  -3.2800
   2.6748
   0.3026 + 1.0238i
   0.3026 - 1.0238i
```

Medida de Tiempo y Esfuerzo de Cálculo

85. El siguiente ejemplo, determina el tiempo requerido por el computador para resolver un sistema lineal de 500 ecuaciones con 500 variables. Ejecute los siguientes comandos
    - `»A=rand(500);b=rand(500,1);x=zeros(500,1)`
    - `»tic;x=A\b;toc`

```
>> tic;x=A\b;toc
Elapsed time is 0.094000 seconds.
```

Integración Numérica de Funciones

86. Cree un archivo .m con el nombre func01.m, que contenga los siguientes comandos:
    - `function y=prueba(x)`
    - `y=1./((x-.3).^2+.01)+1./((x-.9).^2+.04)-6;`
87. Para calcular la integral de func01 entre 0 y 1 utilizando el método de Simpson (cuadratura adaptativa), ejecute el comando `»area=quad('func01',0,1)`

Ecuaciones no Lineales y Optimización

88. Para calcular la raíz de una ecuación no lineal, utilice el comando `»fzero('func01',-.5)`
89. Ejecute el comando `»fzero('func01',2)`. La raíz encontrada depende del segundo argumento, que le da un punto a partir del cual iniciar la búsqueda

```
>> fzero('func01',-.5)

ans =

   -0.1316

>> fzero('func01',2)

ans =

    1.2995
```

Integración Numérica de Ecuaciones Diferenciales Ordinarias

MATLAB es capaz de calcular la evolución en el tiempo de sistemas de ecuaciones diferenciales ordinarias de primer orden, lineales y no lineales. Las ecuaciones diferenciales se pueden escribir en la forma:

$$\dot{y} = f(y,t)$$

Donde *t* es la variable escalar, y tanto *y* como su derivada son vectores. Un ejemplo típico puede ser el lanzamiento parabólico, considerando una resistencia del aire proporcional al cuadrado de la velocidad. Dicha fuerza responde a la expresión vectorial:

$$\begin{Bmatrix} F_x \\ F_y \end{Bmatrix} = -c\sqrt{\left(\dot{x}^2 + \dot{y}^2\right)}\begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix}$$

Donde *c* es una constante conocida. Las ecuaciones diferenciales del movimiento serán:

$$\begin{Bmatrix} \ddot{x} \\ \ddot{y} \end{Bmatrix} = \begin{Bmatrix} 0 \\ -g \end{Bmatrix} - \frac{c}{m}\sqrt{\left(\dot{x}^2 + \dot{y}^2\right)}\begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix}$$

Pero este es un sistema de 2 ecuaciones diferenciales de orden 2. Para poder integrarlo, debe tener la forma de la primera ecuación mostrada en esta serie; para ello se va a transformar en 4 ecuaciones de primer orden de la forma siguiente:

$$\begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{x} \\ \dot{y} \end{Bmatrix} = \begin{Bmatrix} 0 \\ -g \\ u \\ v \end{Bmatrix} - \frac{c}{m}\sqrt{\left(u^2 + v^2\right)}\begin{Bmatrix} u \\ v \\ 0 \\ 0 \end{Bmatrix}$$

MATLAB dispone de varias funciones para integrar sistemas de ecuaciones diferenciales ordinarias de primer orden, entre ellas *ode23*, que utiliza el método de Runge-Kutta de segundo/tercer orden y *ode45*, que utiliza el método de Runge-Kutta-Fehlberg de cuarto/quinto orden.

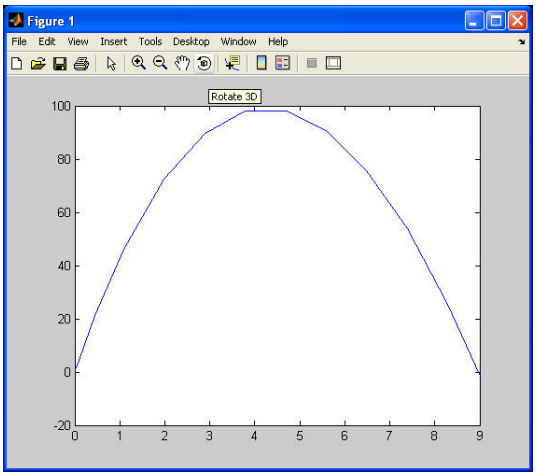
90. Cree una archivo .m con el nombre tiropar.m que contenga los siguientes comandos:

```
function deriv=tiropar(t,y)
fac=-(0.001/1.0)*sqrt((y(1)^2+y(2)^2));
deriv=zeros(4,1);
deriv(1)=fac*y(1);
deriv(2)=fac*y(2)-9.8;
deriv(3)=y(1);
deriv(4)=y(2);
```

Se ha supuesto las constantes con valores c=0.001, m=1 y g=9.8; falta fijar los valores iniciales de posición y velocidad. Se supondrá que el proyectil parte del origen con una velocidad de 100 m/seg y con un ángulo de 30 grados, lo que conduce a los valores iniciales siguientes: u(0)=100*cos(pi/6), v(0)=100*sen(pi/6), x(0)=0 y y(0)=0

91. Ejecute los siguientes comandos:

```
»t0=0;tf=9;
»y0=[100*cos(pi/6) 100*sin(pi/6) 0 0]';
»[t,Y]=ode23('tiropar',[t0,tf],y0);
»plot(t,Y(:,4))
```



Programación usando MATLAB

92. Cree un archivo .m con el nombre nm112.m y los siguientes comandos:

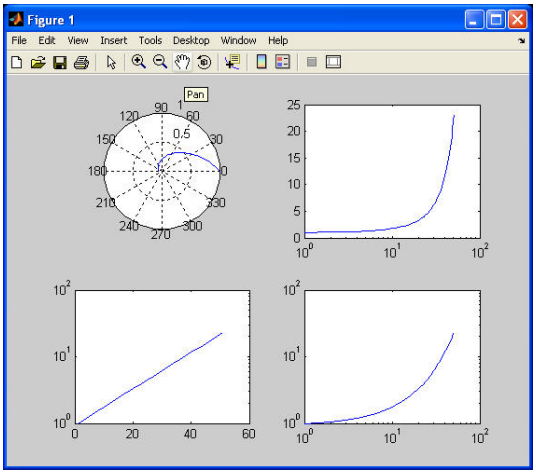
```
%nm112.m
clear
A = [1 2 3;4 5 6]
B = [3;-2;1];
C(2) = 2; C(4) = 4
disp('Presione cualquier tecla para ver el manejo de archivos')
save ABC A B C                % guarda el contenido de A, B y C en el archivo ABC
clear('A','C')                % borra las variables A y C
load ABC A C                  % recupera las variables A y C del archivo ABC
save b.dat B /ascii           % guarda la variable B en el archivo b.dat
clear B                       % borra la variable B
load b.dat                    % recupera la variable B del archivo b.dat
b
x = input('Introduzca x:')    % pide un valor por pantalla
format short e
x
format rat, x
format long, x
format short, x
```

93. Ejecute el comando `»nm112`
94. Cree un archivo .m con el nombre nm114.m y los siguientes comandos:

```
%nm114: diferentes tipos de gráficos
th = [0: .02:1]*pi;
```

```
subplot(221), polar(th,exp(-th))
subplot(222), semilogx(exp(th))
subplot(223), semilogy(exp(th))
subplot(224), loglog(exp(th))
pause, clf
subplot(221), stairs([1 3 2 0])
subplot(222), stem([1 3 2 0])
subplot(223), bar([2 3; 4 5])
subplot(224), barh([2 3; 4 5])
pause, clf
y = [0.3 0.9 1.6 2.7 3 2.4];
subplot(221), hist(y,3)
subplot(222), hist(y,0.5 + [0 1 2])
```

95. Ejecute el comando `»nm114`



96. Cree un archivo .m con el nombre gauss.m y los siguientes comandos:

```
function x = gauss(A,B)
% Los tamaños de las matrices A y B deben ser NA x NA y NA x NB respectivamente
% La función resuelve Ax = B utilizando el algoritmo de eliminación de Gauss
NA = size(A,2); [NB1,NB] = size(B);
if NB1 ~= NA, error('A y B deben tener dimensiones compatibles'); end
N = NA + NB; AB = [A(1:NA,1:NA) B(1:NA,1:NB)];
epss = eps*ones(NA,1);
for k = 1:NA
[akx,kx] = max(abs(AB(k:NA,k))./ ...
max(abs([AB(k:NA,k + 1:NA) epss(1:NA - k + 1)]'))');
if akx < eps, error('Matriz singular; no existe solución única'); end
```

```
mx = k + kx - 1;
if kx > 1 % Cambio de fila si es necesario
tmp_row = AB(k,k:N);
AB(k,k:N) = AB(mx,k:N);
AB(mx,k:N) = tmp_row;
end
% Eliminación de Gauss
AB(k,k + 1:N) = AB(k,k+1:N)/AB(k,k);
AB(k,k) = 1; %Hace que los elementos de la diagonal sean 1
for m = k + 1: NA
AB(m,k+1:N) = AB(m,k+1:N) - AB(m,k)*AB(k,k+1:N); %Eq.(2.2.5)
AB(m,k) = 0;
end
end
x(NA,:) = AB(NA,NA+1:N);
for m = NA-1: -1:1
x(m,:) = AB(m,NA + 1:N)-AB(m,m + 1:NA)*x(m + 1:NA,:);
end
```

97. Ejecute los siguientes comandos:

```
»A=[0 1 1;2 -1 -1;1 1 -1];b=[2 0 1]'
»Gauss(A,b)
```

98. Para comprobar si la solución es la correcta, ejecute el siguiente comando

```
»x1=A\b
```