MATLAB® Conceptos Básicos y Programación

TUTORIAL

Luis Rodríguez Ojeda
lrodrig@espol.edu.ec

Instituto de Ciencias Matemáticas
Escuela Superior Politécnica del Litoral
Guayaquil, Ecuador
2007

MATLAB® Conceptos Básicos y Programación: Tutorial

1 INTRODUCCIÓN

1.1 Objetivo

Proporcionar a los interesados los conocimientos básicos para usar el entorno de MATLAB y las facilidades para su programación.

1.2 Metodología

Mediante explicaciones basadas en los ejemplos incluidos en este manual, el interesado puede adquirir en forma progresiva y autónoma los conocimientos básicos para utilizar MATLAB.

Para progresar rápidamente, puede abrir dos ventanas en la pantalla de su computador, una con el programa MATLAB y otra con este manual, entonces puede escribir y probar cada ejemplo en la ventana de comandos de MATLAB.

1.3 El programa MATLAB

MATLAB (**Mat**rix **Lab**oratory) es un programa interactivo de uso general. Es un instrumento computacional simple, versátil y de gran poder para aplicaciones numéricas, simbólicas y gráficas y contiene una gran cantidad de funciones predefinidas para aplicaciones en ciencias e ingeniería.

La interacción se realiza mediante instrucciones (denominadas comandos), y también mediante funciones y programas en un lenguaje estructurado. Los objetos básicos con los cuales opera MATLAB son matrices. La asignación de memoria a cada variable la realiza MATLAB en forma dinámica y eficiente, por lo que no son necesarias las declaraciones de variables antes de su uso.

1.4 Características de MATLAB

- Cálculo numérico rápido y con alta precisión
- Capacidad para manejo matemático simbólico
- Funciones para graficación y visualización avanzada
- Programación mediante un lenguaje de alto nivel
- Soporte para programación estructurada y orientada a objetos
- Facilidades básicas para diseño de interfaz gráfica
- Extensa biblioteca de funciones
- Paquetes especializados para algunas ramas de ciencias e ingeniería

Operación

- Simple y eficiente
- Interactivo y programable
- Sistema de ayuda en línea
- Interacción con otros entornos

1.5 Uso interactivo de MATLAB

El entorno de MATLAB está organizado mediante ventanas. Las principales son

Command Window. Es la ventana de comandos para ineractuar con MATLAB
Command History. Contiene el registro de los comandos que han sido ingresados.
Workspace. Contiene la descripción de las variables usadas en cada sesión.

Se sugiere al inicio dejar activa únicamente la ventana de comandos, cerrando las otras ventanas. Para restaurarlas use la opción **view** de la barra de herramientas de MATLAB.

El símbolo **>>** indica que el programa está listo para recibir sus instrucciones (comandos)

Ejemplo. Para calcular

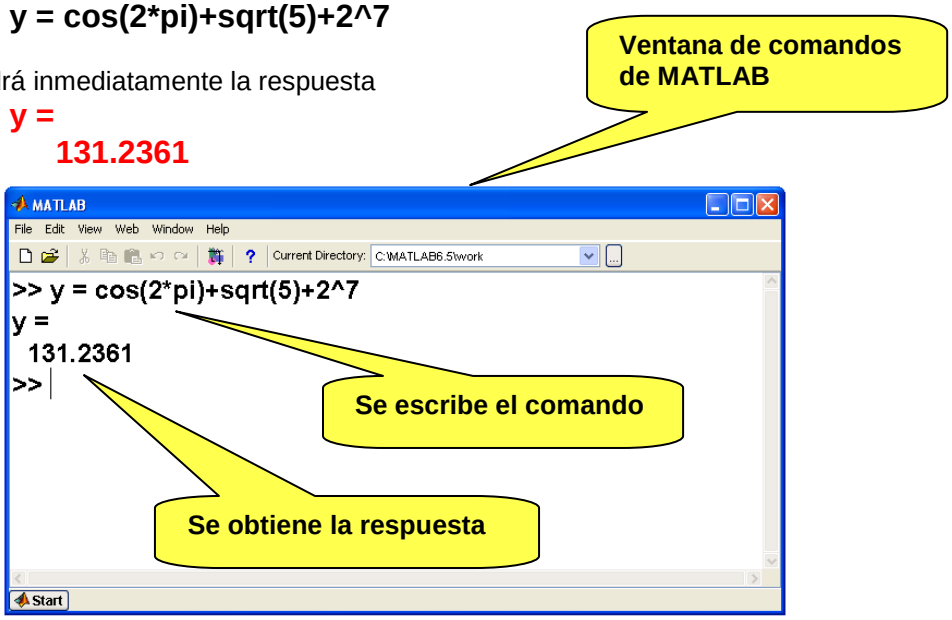
$$y = \cos(2\pi) + \sqrt{5} + 2^7$$

Digite en la ventana de comandos de MATLAB

```
y = cos(2*pi)+sqrt(5)+2^7
```

Obtendrá inmediatamente la respuesta

```
y =
    131.2361
```



1.6 Práctica con comandos de MATLAB

En esta sección se revisa el uso de los comandos principales de MATLAB comenzado con los más elementales. Debe escribir cada ejemplo y presionar la tecla de ingreso.

MATLAB mostrará el resultado inmediatamente, o un mensaje si hubo algún error. Recuerde que la mejor manera de aprender es practicando.

En la mayoría de los ejemplos no se han escrito los resultados que produce MATLAB para evitar que este tutorial sea innecesariamente extenso. Los resultados los puede observar al probar cada comando.

`>> exp(2)/3`	calcule $e^2/3$ y muestre inmediatamente el resultado
`ans = 2.4630`	respuesta mostrada por MATLAB, **ans** significa **answer**
`>> x = exp(2)/3`	calcule $e^2/3$ y asigne el resultado a la variable x
`x = 2.4630`	respuesta mostrada por MATLAB
`>> y = exp(2)/3;`	el ; evita que el resultado se muestre inmediatamente
`>> y`	Escribir la variable para conocer su contenido
`y = 2.4630`	respuesta mostrada por MATLAB
`>> u = 2*x+1`	puede usar el contenido de las variables
`u = 5.9260`	respuesta mostrada por MATLAB
`>> x = x+1`	puede modificar el contenido de las variables
`x = 3.4630`	respuesta mostrada por MATLAB

1.7 Reutilización de comandos

Puede reutilizar comandos presionando las teclas del cursor ↑ ↓

```
>> x = exp(2)/3; y=2*x+1, z=3*x        Puede escribir y ejecutar varios
                                       comandos en una misma línea
   y =
       5.9260                          respuestas mostradas por MATLAB
   z =
       7.3891
```

1.8 El sistema de ayuda de MATLAB

MATLAB ofrece una descripción detallada del uso de cada comando y cada función digitando **help** y el nombre del comando.

Ejemplo. Para conocer el uso de la función **sqrt**, digite

```
>> help sqrt
```

```
SQRT   Square root.                                   Esta información
SQRT(X) is the square root of the elements of X. Complex   aparece en pantalla
results are produced if X is not positive.
```

```
>> help          despliega temas de ayuda
>> help ops      despliega comandos de un tema.  Ej.  lista de operadores
>> help exp      uso de un comando específico.   Ej. función exponencial
```

Adicionalmente, presionando el ícono **Help** usted puede entrar al sistema de ayuda de MATLAB organizado por contenido, índice, búsqueda y demostraciones.

1.9 Algunos ejemplos para practicar en MATLAB

Digite cada uno de los siguientes ejemplos en la ventana de comandos. Al final de cada ejemplo se ha escrito con **letra azul** una breve explicación para faciltar la comprensión de cada comando y el resultado que se obtendrá.

1) Para resolver el sistema:

$$2x + 3y = 4$$
$$5x - 2y = 6$$

Digite en la ventana de comandos

```
>> a = [2, 3; 5, -2];          Ingresar la matriz de coeficientes
>> b = [4; 6];                 Ingresar el vector columna de constantes
>> x = inv(a)*b;               Obtener la solución invirtiendo la matriz
   x = 1.3684                  Vector solución
       0.4211
```

2) Integrar la función f(x) = x sen(x), evaluar el integral, derivar

```
>> f = 'x*sin(x)';
>> h = int(f)                  Integrar analíticamente
   h =  sin(x)-x*cos(x)
>> r = eval(int(f, 0, 2))      Evaluar el Integral entre 0 y 2
   r = 1.7416
>> g = diff(f)                 Primera derivada de f(x)
   g = sin(x)+x*cos(x)
```

3) Para resolver la ecuación cúbica $5x^3 + 2x^2 - 3x + 1 = 0$;

```
>> a = [5, 2, -3, 1];          Ingresar los coeficientes de la ecuación
>> x = roots(a)                Obtener las tres raíces
   x =  -1.1060                Una raíz real y dos raíces complejas
        0.3530 + 0.2371i
        0.3530 - 0.2371i
```

4) Para obtener la solución de la ecuación diferencial ordinaria: $y' - x - y = 0$, $y(0) = 1$

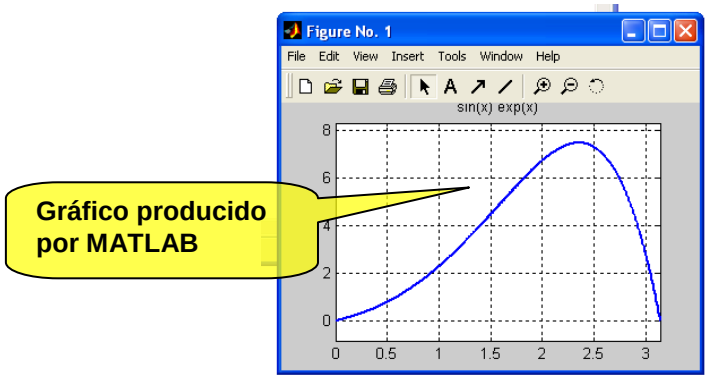
```
>> y = dsolve('Dy-x-y=0','y(0)=1', 'x')      Definir la ecuación, condición y variable
   y = -x-1+2*exp(x)                        Solución analítica
```

5) Manejo simbólico de expresiones

```
>> syms x                                   Definir x con tipo simbólico
>> y = x^3- 8;                              La expresión x^3 – 8 se asigna a y
>> t = factor(y)                            Factorar la expresión
   t = (x-2)*(x^2+2*x+4)
>> t = taylor(exp(x), 5);                   Expandir e^x con 5 términos de la serie de
   t = 1+x+1/2*x^2+1/6*x^3+1/24*x^4         Taylor
```

6) Para graficar la función $f(x)=sen(x)\,e^x$ en el intervalo $0 \le x \le \pi$

```
>> f = 'sin(x)*exp(x)';                     Escribir la función entre comillas simples
>> ezplot(f, [0, pi]);                      Función para graficar
>> grid on;                                 Mostrar cuadrículas en el gráfico
```



1.10 Símbolos especiales en MATLAB

[ ]	para definir vectores y matrices
( )	para definir precedencia en expresiones y para subíndices
,	para separar elementos de un vector use comas o espacios
;	para separar filas y para evitar mostrar contenido de variables
%	para iniciar un comentario (programas y funciones)
...	para continuar un comando en la siguiente línea

2 CÁLCULO NUMÉRICO

2.1 Formatos de exhibición de números en la pantalla

```
>> format long                  muestra 14 decimales
>> x=exp(2)                     un ejemplo para visualizar
   x =
       7.38905609893065
>> format bank                  formato para 2 decimales
>> x
   x =
       7.39
>> format rat                   notación racional (fracciones)
>> x
   x =
       2431/329
>> format short e               notación científica
>> x
   x =
       7.3891e+000
```

```
>> format long e              notación científica con 14 decimales
>> format +                   muestra signos +,  , -
>> format short               4 decimales  (MATLAB lo usa por omisión)
>> format compact             suprime líneas adicionales en la salida
>> format loose               inserta líneas en blanco en la salida(recomendado)
>> format hex                 formato hexadecimal
>> vpa(sqrt(2), 20)           variable precision arithmetic
   ans =                      (muestra la raíz cuadrada de 2 con 20 dígitos
        1.4142135623730950488
>> format short               regrese al formato normal de MATLAB
```

2.2 Operadores aritméticos

```
+ - * / \ ^ ( )        ^  se usa para potenciación
                       /  es división a la derecha
                       \  es división a la izquierda (para matrices)
>> help ops            listar los operadores y caracteres especiales
```

2.3 Funciones matemáticas

```
>> help elfun          listar las funciones matemáticas elementales
      Trigonometric.
      sin       - Sine.
      sinh      - Hyperbolic sine.
      asin      - Inverse sine.
      asinh     - Inverse hyperbolic sine.
       . . .
```

2.4 Operadores relacionales y lógicos

```
< <= > >= == ~= & | ~      los tres últimos corresponden a: ∧ ∨ ¬
                           == representa al símbolo =
                           ~= representa al símbolo ≠

>> t=sin(2) < 0.8 & log(2) > 0.5      el resultado es un valor lógico  (0  o  1)
```

2.5 Símbolos numéricos especiales

```
>> 2/0
        Inf                es el símbolo ∞
>> 0/0
        NaN                significa “Not A Number” (valor indeterminado)
>> pi                      contiene la constante π
>> eps                     es la precisión del tipo real en MATLAB
>> realmin                 el menor número real en MATLAB
>> realmax                 el mayor número real en MATLAB
```

2.6 Manejo de números complejos

i representa al símbolo $\sqrt{-1}$

```
>> x = 3+2i                asignar un número complejo
>> t = 2*x + 3 - 5i        operación con números complejos
   t =
      9.0000 - 1.0000i
>> y =exp(x)               el resultado también es complejo
   y =
     -8.3585 +18.2637i
>> y = log(-2)             el referencial de MATLAB son los complejos
   y =
      0.6931 + 3.1416i
```

2.6.1 funciones para números complejos

conj, real, imag, abs, angle, complex

```
>> z=3+2i;
>> t=conj(z)               obtener el conjugado
```

3 VARIABLES

- No requieren ser declaradas
- Su tipo depende del valor asignado
- Pueden ser redefinidas
- Sensible al tipo de letra (mayúsculas o minúsculas)
- **ans** es la variable por omisión provista por MATLAB
- MATLAB realiza la asignación de memoria a variables durante la ejecución.

>> x=3	x es de tipo real
>> x='mensaje'	ahora x es de tipo literal (use comillas simples)
>> syms x	x se redefine a tipo símbolo
>> x=[2 7 4]	x es ahora un vector un vector
>> x=2+3i	x es de tipo complejo
>> x	muestre el contenido actual de la variable
>> whos x	muestre el tipo actual de la variable
>> disp(x)	muestre solamente el contenido
>> x=input('¿dato?');	ingrese un valor para una variable desde el teclado
>> exp(x)/3	
>> ans	la variable **ans** contiene el último resultado
>> y=2*ans	la puede usar

4 ALGUNOS COMANDOS DEL SISTEMA OPERATIVO

>> help general	lista de comandos
>> who	lista las variables en uso
>> whos	lista las variables en uso y su descripción
>> exist('c')	chequea si la variable **c** existe
>> clear a b c	**clear** borra variables
>> clc	despeja la ventana de comandos
>> pwd	muestra cual es el directorio actual
>> cd c:\MATLAB\work	**cd** cambia la ruta del directorio actual
>> dir	lista el contenido del directorio actual
	También se lo puede hacer con las opciones
	de la barra de herramientas
>> save prueba	**save** almacena las variables en un archivo
>> load prueba	**load** carga variables y su contenido ejemplo
>> delete prueba.mat	**delete** elimina archivo ejemplo
>> quit	para terminar la sesión con MATLAB (no lo digite aun)

4.1 Comandos especiales

>> date	fecha
>> clock	fecha hora, vea su uso con help.
>> format rat	para visualizar la fecha con mas claridad
>> clock	
>> format short	vuelva al formato normal

5 CADENAS DE CARACTERES

>> x='Matematica';	asignación de una cadena (use comillas simples)
>> x(4)	manejo de un carácter de la cadena, use un indice
	En MATLAB los índices se escriben entre paréntesis y son numerados desde 1
>> t=x(2:5);	manejo de una subcadena, use: **nombre(inicio: final)**
>> n=length(x)	longitud de la cadena
>> c=strcat(x, t)	concatenación de cadenas
>> help strfun	listar las funciones para cadenas

6 VECTORES Y MATRICES

```
>> x=[3, -1, 4, 7, -2]
```
asignación directa de un vector fila

```
>> x=[3  -1  4  7  -2]
```
puede separar con **comas** o con **espacios**

```
>> x(2)=5
```
manejo de un componente del vector.

En MATLAB los índices se escriben entre paréntesis y son numerados desde 1

```
>> y=x(2: 4)
```
para asignar parte de un vector use **(inicio: final)**

```
   y =
       -1    4    7
```

```
>> t=[3; -1; 4; 5]
```
para asignar un vector columna use **;**

```
   t =
        3
       -1
        4
        5
```

```
>> t=x'
```
para obtener la transpuesta de un vector use **'**

x' es la transpuesta del vector **x**

```
>> y = [3, x, -6, 7]
```
puede asignar un vector usando otro vector

```
   y =
     3    3   -1    4    7   -2   -6    7
```

```
>> y = 2:1:10
```
puede asignar un vector mediante una secuencia

```
   y =
     2    3    4    5    6    7    8    9   10
```

En MATLAB las secuencias se escriben: valor inicial : incremento : valor final
si el incremento es 1 puede omitirlo

```
>> y=[2, 5, 4, ...
       7, -3]
```
Para continuar en la siguiente línea use **...**
Escribir la continuación de la línea anterior

```
>> x=[3, 5, 2, 0]
>> y=2*x
>> y=exp(x)
```
puede realizar operaciones escalares
o crear vectores con funciones

```
>> a = [6  3 ; 5  1]
   a =
     6    3
     5    1
```
asignación directa de una matriz 2x2

separe elementos con espacios o comas
separe filas con punto y coma

```
>> a(2,1)
```
manejo de los componentes de una matriz con índices numerados desde 1: **(fila, columna)**

```
>> a=[2, -3; 5, 1; 0, 7]
>> x=[7, 3]
>> a=[x; x]
>> b=[5, 6]
>> c=[a; b]
>> d=[a, b']
>> x=c(1, :)
>> x=c(: ,1)
>> c(:,2)=[ ]
```
una matriz 3x2

una matriz 2x2

c es una matriz aumentada 3x2
c es una matriz aumentada 2x3
asigne a **x** la primera fila de **c**
asigne a **x** la primera columna de **c**
elimine la segunda columna de **c**

6.1 Matrices especiales

```
>> a=ones(3)
>> a=ones(3,5)
>> a=zeros(4,5)
>> a=eye(5)
>> a=magic(4)
>> a=hilb(5)
```
matriz 3x3 iniciada con unos
matriz 3x5 iniciada con unos
matriz 4x5 iniciada con ceros
matriz identidad 5x5
cuadrado mágico 4x4
matriz de Hilberth 5x5

```
>> x=[2, 5, 3, 7];          un vector
>> a=vander(x)              matriz de Vandermonde 4x4 usando un vector
>> a=[ ]                    matriz nula
```

6.2 Una matriz puede componerse con otras matrices

```
>> a = rand(3);             matriz 3x3 con números aleatorios
>> b = [5 3 9];             vector de tres componentes
>> e = diag(b);             matriz 3x3 con b en la diagonal
   e =
      5    0    0
      0    3    0
      0    0    9
>> c=eye(3);                matriz identidad 3x3
>> d=zeros(3);              matriz con ceros 3x3
>> t=[a e; c d]             matriz compuesta 9x9
```

6.3 Editor de vectores y matrices

En la ventana **workspace** puede activar el editor de arreglos, similar a una hoja electrónica, con el cual puede modificar con facilidad las dimensiones y el contenido de vectores y matrices.

6.4 Elementos de vectores y matrices pueden manejarse con otro vector o matriz

```
>> x=[ 8 7 9 5 6];
>> p=[2 4 1];               vector para direccionar al vector  x
>> t=x(p)                   t  contiene los elementos 2, 4 y.1 del vector x
>> a=[4 7 3; 5 7 8; 6 0 9];
>> p=[1 3];                 vector para direccionar las filas de la matriz  a
>> q=[2 3];                 vector para direccionar las columnas de la matriz a
>> t=a(p, q)                t  contiene las filas 1 y 3, columnas 2 y 3 de a
```

6.5 Operaciones con matrices

```
>> a=[3, 2; 1, 4];
   a =
      3    2
      1    4
>> b=[8, 6; 5, 7];
>> c=a'                     transpuesta de a
   c =
      3    1
      2    4
>> c=2*a                    producto de un escalar por matriz
   c =
       6    4
       2    8
>> c=a+b                    suma de matrices
   c =
      11    8
       6   11
>> c=a*b                    producto de matrices
   c =
      34   32
      28   34
>> c=a.*b                   producto elemento por elemento de matrices
   c =                              para operar elemento a elemento use un punto
      24   12                       antes del operador
       5   28

>> c=a^2                    matriz al cuadrado, equivale a:  a*a
>> c=a.^2                   cada elemento de la matriz a, elevar al cuadrado
>> c=a==b                   compare igualdad entre matrices (de igual tamaño)
```

el resultado es una **matriz binaria** (ceros y unos)

>> c=a~=b compare si dos matrices no son iguales
el resultado es una **matriz binaria** (ceros y unos)

>> c=a>3 compare si cada elemento de **a** es mayor a 3
el resultado es una **matriz binaria** (ceros y unos)

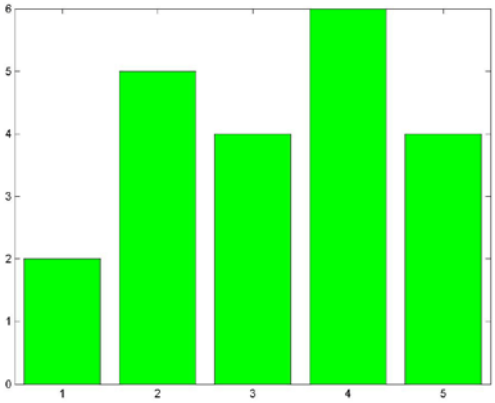
6.6 Funciones para operar con matrices

```
>> x=[-2, 0, 6, 5];          un vector para los ejemplos
>> a=[1, 2, 3; 4, 5, 6;7, 8, 9];   una matriz para los ejemplos
>> n=length(x)               longitud del vector x
>> [n,m]=size(a)             tamaño de la matriz a: el resultado es un vector
>> n                         número de filas: 3
>> m                         número de columnas: 3
>> isempty(a)                chequea si un vector o matriz está vacío
>> any(x)                    determina si el vector contiene algún valor no cero
>> any(a)                    igual que arriba, pero por columnas de la matriz
>> t=find(x)                 obtiene índices de elementos del vector no ceros
>> t=find(x>3)               obtiene los índices de cada elemento > 3
>> [f,c]=find(a)             obtiene los índices de filas y columnas de la matriz
                             cuyos elementos son no ceros

>> t=dot(x, x)               producto punto entre dos vectores
>> k=rank(a)                 rango de a
>> t=trace(a)                traza de a
>> d=det(a)                  determinante de a
>> b=inv(a)                  inversa de a
>> h=norm(a, 1)              norma de columna de la matriz a
>> h=norm(a, inf)            norma de fila de la matriz a
>> h=norm(x, inf)            norma de fila o columna del vector x
>> c=cond(a)                 número de condición de la matriz a
>> t=diag(a)                 vector con la diagonal de la matriz a
>> t=diag(x)                 matriz con x en la diagonal
>> t=rot90(a)                rote a 90 grados (sentido opuesto al reloj)
>> t=fliplr(a)               voltee horizontalmente la matriz a
>> t=tril(a)                 obtenga la matriz triangular inferior de a
>> t=triu(a)                 obtenga la matriz triangular superior de a
>> b=[5,-1; 3, 4; 2, 7];
>> b=reshape(b, 2, 3)        reconfigura la matriz b de 3x2 a 2x3

>> [t,s]=lu(a)               descomposición triangular de a en las matrices
>> t                         triangulares t y s tales que t*s es igual que a
>> s
>> t*s                       se obtiene la matriz a
>> t=cov(a)                  matriz de covarianza de a
>> e=eig(a)                  valores propios de a
>> p=poly(a)                 polinomio característico de a
>> r=roots(ans)              valores propios de a
>> help matfun               liste las funciones para matrices
```

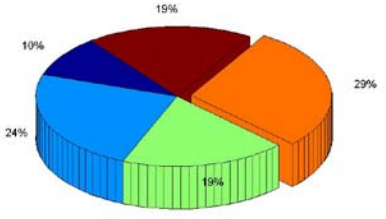
6.7 Funciones adicionales para manejo de datos con vectores y matrices

```
>> x=[2, 5, 4, 6, 4];        un vector
>> a=[5,-1; 3, 4; 2, 7];     una matriz
>> t=max(x)                  el mayor valor del vector x
   t =
      6
>> v=max(a)                  el mayor valor por columnas de la matriz a
   v =
      5     7
>> t=sum(x)                  suma de componentes
>> v=sum(a)                  suma de componentes por columnas
```

>> t=prod(x)	producto escalar
>> v=prod(a)	producto escalar por columnas
>> t=cumsum(x)	suma acumulada
>> v=cumsum(a)	suma acumulada por columnas
>> t=cumprod(x)	producto acumulado
>> v=cumprod(a)	
>> t=mean(x)	media aritmética
>> v=mean(a)	
>> t=median(x)	mediana
>> v=median(a)	
>> t=std(x)	desviación estándar
>> v=std(a)	
>> t=sort(x)	ordenamiento ascendente
>> v=sort(a)	
>> t=dsort(x)	ordenamiento descendente
>> bar(x)	diagrama de barras



>> bar(a)	
>> hist(x)	histograma
>> stairs(x)	dibuja x mediante escalones
>> pie(x)	gráfico tipo pastel
>> pie3(x)	pastel en relieve
>> v=[0,0,0,1,0]	vector para extraer sectores del pastel
>> pie3(x,v)	gráfico tipo pastel 3-d con un sector separado



7 GENERACIÓN DE NÚMEROS ALEATORIOS

>> x=**rand**	genera un número aleatorio entre 0 y 1
>> a=**rand**(5)	genera una matriz 5x5 con números aleatorios
>> b=**rand**(4,5)	genera una matriz 4x5 con números aleatorios
>> d=fix(rand*10)+1	transfomación para obtener un entero aleatorio entre 1 y 10

8 INGRESO DE PUNTOS DESDE LA PANTALLA CON EL MOUSE

```
>> ezplot('sin(x)');          ejemplo para tomar puntos desde un gráfico
>> grid on
>> [x,y]=ginput(5);           ingrese 5 puntos desde la pantalla .
                              Presione el botón del mouse para
                              ingresar cada punto
>> x                          observe las abscisas
>> y                          y las ordenadas ingresadas
>> plot(x, y, 'o')            grafique los puntos ingresados
```

9 POLINOMIOS

```
>> a=[2, -3, 0, 5],           define el polinomio  2x^3 – 3x^2 + 5
>> y=polyval(a,4)             evaluación del polinomio con un valor
>> x=roots(a)                 obtenga un vector con raíces (reales y complejas)
>> t=polyval(a, x(1))         verifique una raíz
>> p=poly(x)                  producto de todas las raíces
>> b=[3, 4, -2];              define el polinomio  3x^2+ 4x - 2
>> c=conv(a,b)                producto de polinomios
>> [c, r]=deconv(a,b);        división de polinomios
>> c                          cociente
>> r                          residuo
>> x=[2 3 5 7 8];             abscisas de puntos (x,y)
>> y=[3.2 4.1 5.8 6.4 6.3];   ordenadas de los puntos
>> z=3.2;                     valor para interpolar, z puede ser un vector
>> u=interp1(x,y,z,'linear')  resultado de la interpolación lineal
>> u=spline(x,y,z)            interpolación con un trazador cúbico
>> a=polyfit(x, y, 2);        polinomio de mínimos cuadrados de grado 2
>> a                          el vector a  contiene los coeficientes
```

10 MANEJO SIMBÓLICO

```
>> syms x;                    definición de variable tipo simbólico
>> 2*x+3*x                    suma algebraica
>> a=[x 5; 3*x 4];            matriz con elementos símbolos
>> t=inv(a)                   su inversa también contiene símbolos
>> f=3*x^2+5*x;               definición simbólica de una función
>> t=factor(f)                factorar la expresión
>> s=expand(t)                expandirla
>> e=taylor(exp(x))           expansión con la serie de Taylor
>> limit(sin(x)/x)            obtencíon de límites de funciones
>> syms y;
>> f=2*x^3+3*y^2              una función de dos variables
>> g=diff(f,x)                derivada parcial
>> u=int(f,x)                 integrar en x
>> f='2*t+1';                 definición de una función en forma literal
>> t=3;
>> y=eval(f)                  evaluación de la función
```

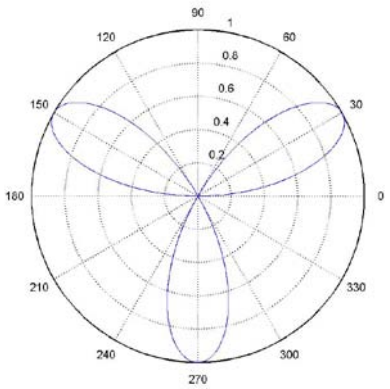
11 FUNCIONES ESPECIALES PARA MEDIR EFICIENCIA DE ALGORITMOS

```
>> tic;                       Inicia cronómetro
>> toc;                       muestra el tiempo transcurrido

>> tic; a=inv(rand(500, 500)); toc
                              tiempo utilizado en invertir una matriz 500x500
```

12 GRAFICACIÓN

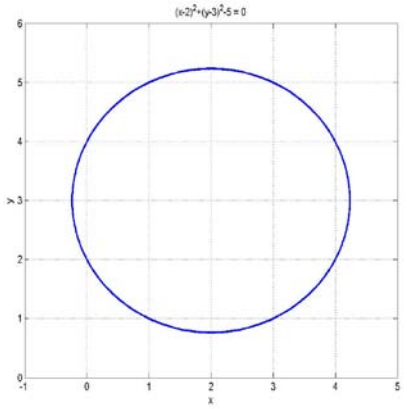
12.1 Gráfico de funciones de una variable

```
>> f='exp(x)-3*x';                 función para el ejemplo (use comillas simples)
>> ezplot(f)                       función básica para graficar f en [-2π, 2π]
>> ezplot(f, [0, 2])               función básica para graficar f en un dominio dado
>> grid on                         colocar cuadrículas en el dibujo

>> x=[0: 0.1: 2*pi];               puntos para evaluar alguna función
>> y=sin(x);                       puntos de la función seno
>> plot(x,y);                      función para graficar la función con línea contínua
>> plot(x,y,'o')                   gráfico con puntos. Puede elegir: o . * + x --
>> plot(x,y,'r')                   cambiar a color rojo. Puede elegir  r,b,y,m,g,w,k
>> plot(x,y,'og')                  grafique con círculos verdes.
>> grid on                         colocar cuadrículas en el dibujo

>> title('seno de x')              incluya un título en el gráfico
>> gtext('seno de x')              posicione el texto en el gráfico con el mouse
>> xlabel('X')                     rotule el eje horizontal
>> ylabel('Y')                     rotule el eje vertical

>> c=[0, 2*pi, -2, 2]              defina la región para el gráfico
>> axis(c)

>> hold on                         superponer siguientes gráficos
>> hold off                        deshabilitar opción anterior
>> clf                             borrar el gráfico

>> figure(1)                       puede tener varias figuras abiertas
                                   cada una en una ventana rotulada con 1, 2, ...
>> subplot(2,3,1)                  puede dividir una figura en subgráficos.
                                   Ej. en 2 filas y 3 columnas. Activando el gráfico 1
>> clf(1)                          borra el gráfico 1
>> clf                             borre todos los gráficos
>> x=[0:0.1:10};
>> y=exp(x);
>> semilogx(x,y)                   graficar en escalas logarítmicas
>> semilogy(x,y)
>> loglog(x,y)                     doble logarítmica
>> grid on

>> a=0:0.01:2*pi;
>> r=sin(3*a);                     'rosa' de 3 pétalos
>> polar(a, r);                    grafique en coordenadas polares
```

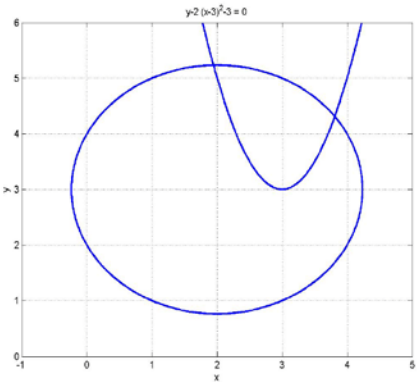


12.2 Gráfico de funciones implícitas y ecuaciones con dos variables

```
>> f='(x-2)^2+(y-3)^2-5';
>> ezplot(f,[-1,5,0,6])        Graficar f en el dominio  -1≤x≤5, 0≤y≤6
>> grid on;                    Colocar cuadrículas
```



```
>> hold on;                    Superponer el siguiente gráfico:
>> g='y-2*(x-3)^2-3';          una parábola y=2(x-3)^2-3  en el mismo dominio
>> ezplot(g,[-1,5,0,6])
```



12.3 Gráfico de funciones definidas en forma paramétrica

```
>> ezplot('sin(t)','cos(t)',[-pi,pi]);          Graficar x=x(t), y=y(t) en -π≤t≤π
>> ezplot('sin(3*t)*cos(t)','sin(3*t)*sin(t)',[0,pi]);   Una rosa de 3 pétalos
```

12.4 Editor de gráficos

Después que el gráfico ha sido realizado puede utilizar las facilidades del editor de gráficos para cambiar las propiedades de las figuras: color, tipo, etc. También puede realizar estadísticas básicas y ajuste de curvas. Adicionalmente puede insertar directamente en el gráfico texto, líneas, flechas, rótulos, etc.

Para habilitar el editor de gráficos seleccione el botón **tools** en la barra de opciones del gráfico y luego elija **edit plot**. Para realizar estadísticas básicas y ajuste de curvas, elija respectivamente **Data Statistics** y **Basic Fitting**

Ejercicio. Obtenga y grafique el polinomio de interpolación, la recta de mínimos cuadrados y el trazador cúbico para un conjunto de datos dados

```
>> x=[1 2 4 5 7];          cinco puntos (x, y) para el ejemplo
>> y=[5 3 6 7 4];
>> plot(x,y,'o')           grafique los datos con círculos
>> grid on                 poner cuadrículas
>> hold on                 superponer los siguientes gráficos

>> a=polyfit(x,y,4);       polinomio de interpolación, 5 puntos: grado 4
>> a                       coeficientes a(1)x^4 + a(2)x^3 + a(3)x^2 + ...
>> z=[1: 0.1: 7];          puntos para evaluar el polinomio
```

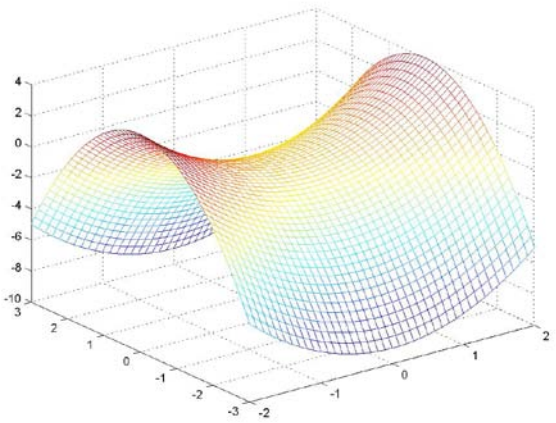
```
>> p=polyval(a,z);          evalúe el polinomio con z  obtenga puntos p
>> plot(z,p)                grafique el polinomio de interpolación

>> b=polyfit(x,y,1);        recta de mínimos cuadrados (grado 1)
>> b                        coeficientes de la recta:  b(1)x + b(2)
>> t=[1 7];                 puntos extremos de la recta (abscisas)
>> q=polyval(b,t);          obtenga las ordenadas respectivas de la recta
>> plot(t,q,'r')           grafique la recta en color rojo

>> s=spline(x,y,z);         evalúe con z  el trazador cúbico y obtenga  s
>> plot(z,s,'g')            grafique el trazador cúbico con verde
>> hold off                 deshabilite la superposición de gráficos
```

12.5 Gráfico de funciones de dos variables

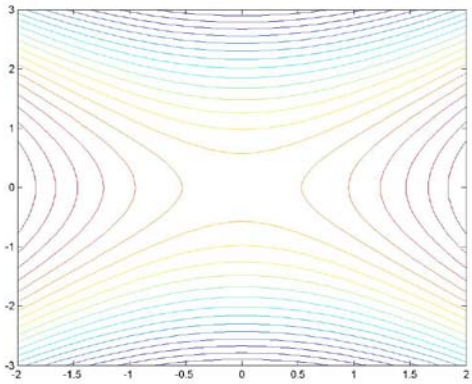
```
>> a=[1 3 2; 5 3 7; 4 5 2];  una matriz 3x3
>> mesh(a);                  graficar los elementos como puntos sobre el plano.
```

El siguiente ejemplo es una referencia para graficar funciones de dos variables
Graficar **$z = x^2 - y^2$, $-2 \le x \le 2$, $-3 \le y \le 3$**

```
>> x=-2:0.1:2;              dominio de la función para el ejemplo
>> y=-3:0.1:3;
>> [u,v]=meshgrid(x,y);      u, v: matrices q' contienen cada par ordenado  x,y
>> z=u.^2 - v.^2;           puntos de la función  z = x^2 – y^2
>> mesh(x, y, z)            gráfico de malla
```



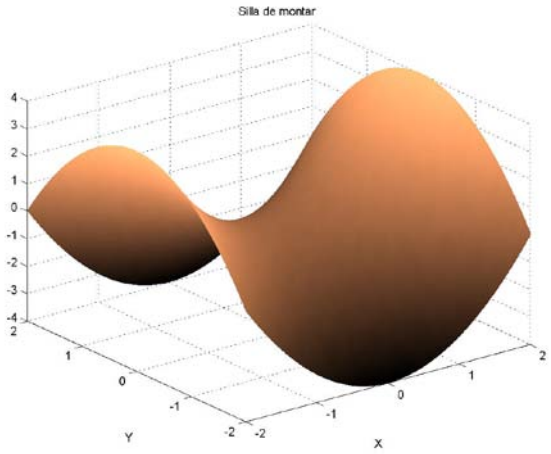
```
>> contour(x, y, z, 20)     gráfico de contorno con 20 curvas de nivel
```



```
>> surfc(x, y, z)           gráfico de superficie y contorno
>> surf(x, y, z)            gráfico de superficie
>> xlabel('X')              rotulación de eje x
```

```
>> ylabel('Y')          rotulación de eje y; también puede usar zlabel
>> title('Silla de montar')   título para el gráfico
>> colormap copper;     color del gráfico; también: gray, jet, pink
>> shading interp;      suavizado del gráfico
```

Gráfico final



Adicionalmente puede usar las opciones del editor de gráficos para editar la figura, rotar, cambiar la perspectiva, insertar títulos, etc.

12.6 Para insertar el gráfico en un documento

Si desea insertar el gráfico elaborado con **MATLAB** en un documento, usualmente escrito en **WORD**, puede seguir el siguiente procedimiento:

1) Elija en la barra de opciones del gráfico el botón **File** y luego la opción **Export**
2) Elija una carpeta para almacenar el gráfico y un nombre para el gráfico.
3) Guarde el gráfico con tipo .**jpg**
4) Copie el gráfico almacenado y péguelo en el documento, en el lugar elegido y reduzca el tamaño hasta encuadrarlo en el texto.

13 FUNCIONES PARA ESPECIALES PARA ANÁLISIS NUMÉRICO

13.1 Raíces de ecuaciones no lineales

```
>> f='exp(x)-pi*x';
>> x=solve(f)
>> x=eval(x)                cambia la solución simbólica a real
       x =
          0.5538            resultados de MATLAB
          1.6385
>> x=fzero(f,2)             solución de una ecuación con un valor inicial
       x =
          1.6385            resultado de MATLAB
>> x=fzero(f,[1,2])         solución usando un rango para la raiz
       x =
          1.6385            resultado de MATLAB
```

13.2 Raíces de sistemas de ecuaciones no lineales

Resolver el sistema: $a^2 + ab - b = 3$, $a^2 - 4b = 5$

```
>> [a,b] = solve('a^2 + a*b - b = 3','a^2 - 4*b = 5');
>> a=eval(a)                para expresar la solución en forma real
   a =
       -1.0000              resultados entregados por MATLAB
        1.8284
       -3.8284
```

```
>> b=eval(b)                          resultados entregados por MATLAB
   b =
        -1.0000
        -0.4142
         2.4142
```

13.3 Integración

```
>> f = 'exp(x)-pi*x';
>> v = int(f)                         integración analítica
   v =
      exp(x)-1/2*pi*x^2
>> r = eval(int(f, 0, 2))             integración entre límites
   r =
        0.1059
>> g = 'x*exp(-x)';
>> r = int(g, 0, Inf);                integral impropia
   r =
        1
```

13.4 Diferenciación

```
>> u = diff(f)                        diferenciación con una variable
   u =
        exp(x)-pi
>> f = 'x*exp(x+y)';
>> u = diff(f,'x')                    diferenciación con dos variables
   u =
        exp(x+y)+x*exp(x+y)
```

13.5 Ecuaciones diferenciales ordinarias de primer orden

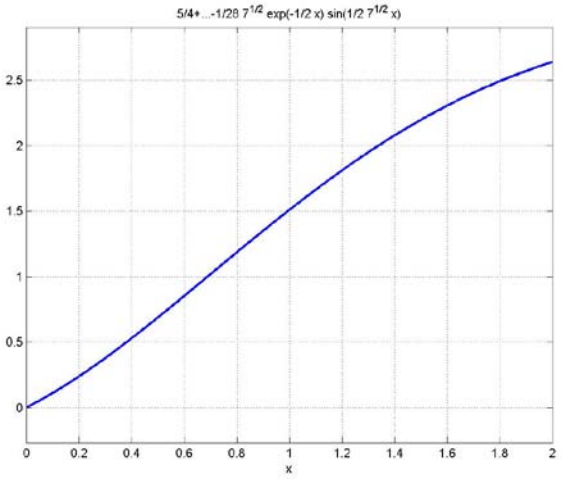
Resolver la ecuación $y' = (x - y)/x,\ \ y(0) = 0$

```
>> y=dsolve('Dy=(x-y)/x','y(0)=0','x')
>> ezplot(y,0,2);
>> grid on
```

13.6 Ecuaciones diferenciales ordinarias de segundo orden con cond. en el inicio

Resolver la ecuación $y'' + y' + 2y - x - 3 = 0, y(0) = 0, y'(0) = 1$

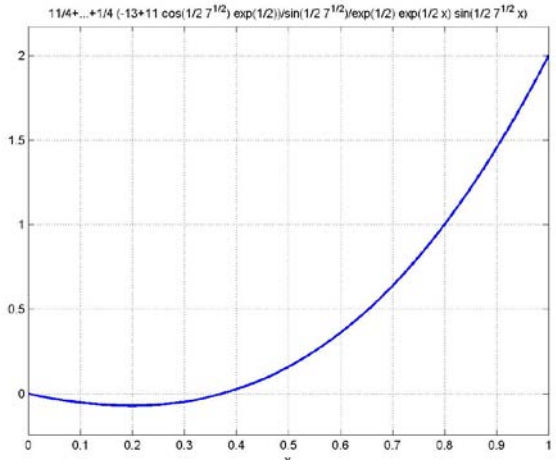
```
>> y=dsolve('D2y+Dy+2*y-x-3=0','y(0)=0,Dy(0)=1','x')
y =                                                    Solución calculada
5/4+1/2*x-5/4*exp(-1/2*x)*cos(1/2*7^(1/2)*x)-1/28*7^(1/2)*exp(-1/2*x)*sin(1/2*7^(1/2)*x)

>> ezplot(y,0,2), grid on
```



13.7 Ecuaciones diferenciales ordinarias de segundo orden cond. en los bordes

Resolver la ecuación $y'' - y' + 2y - 5x - 3 = 0, y(0) = 0, y(1) = 2$

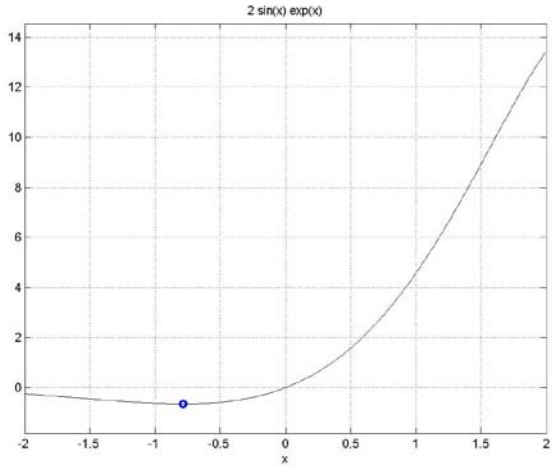
```
>> y=dsolve('D2y-Dy+2*y-5*x-3=0','y(0)=0,y(1)=2','x');
>> ezplot(y, [0, 1])
```



13.8 Optimización

Encontrar un mínimo local de $f(x) = 2\text{sen}(x)e^x$, $-4 \le x \le 4$

```
>> f='2*sin(x)*exp(x)';
>> [x,y]=fminbnd(f,-2,2)
x =
   -0.7854
y =
   -0.6448
>> ezplot(f,-2,2), grid on
>> hold on
>> plot(x,y,'o');
```



Ejercicio. Escribir las instrucciones necesarias para encontrar el valor del radio x de un cilindro de 1000 cc de capacidad, de tal manera que el valor del área sea el mínimo:

Primer enfoque:

1) Escribir una función f en términos del radio x
2) Grafique f con ezplot. Localice el intervalo para el mínimo de f(x)
3) Use la función **fminbnd** para obtener el mínimo
   x: radio, h: altura

```
>> f='2*pi*x*1000/(pi*x^2)+2*pi*x^2';
>> ezplot(f,0,10), grid on
>> x=fminbnd(f,4,6)
x =
   5.4193
>> area=eval(f)
area =
  553.5810
```

Segundo enfoque

1) Derive f y obtenga la función a minimizar g.
2) Grafique g con ezplot. Localice el intervalo de la raíz de g(x)=0
3) Use la función **fzero** para obtener la raíz
4) Use la funcion **solve** para obtener la raíz
   x: radio, h: altura

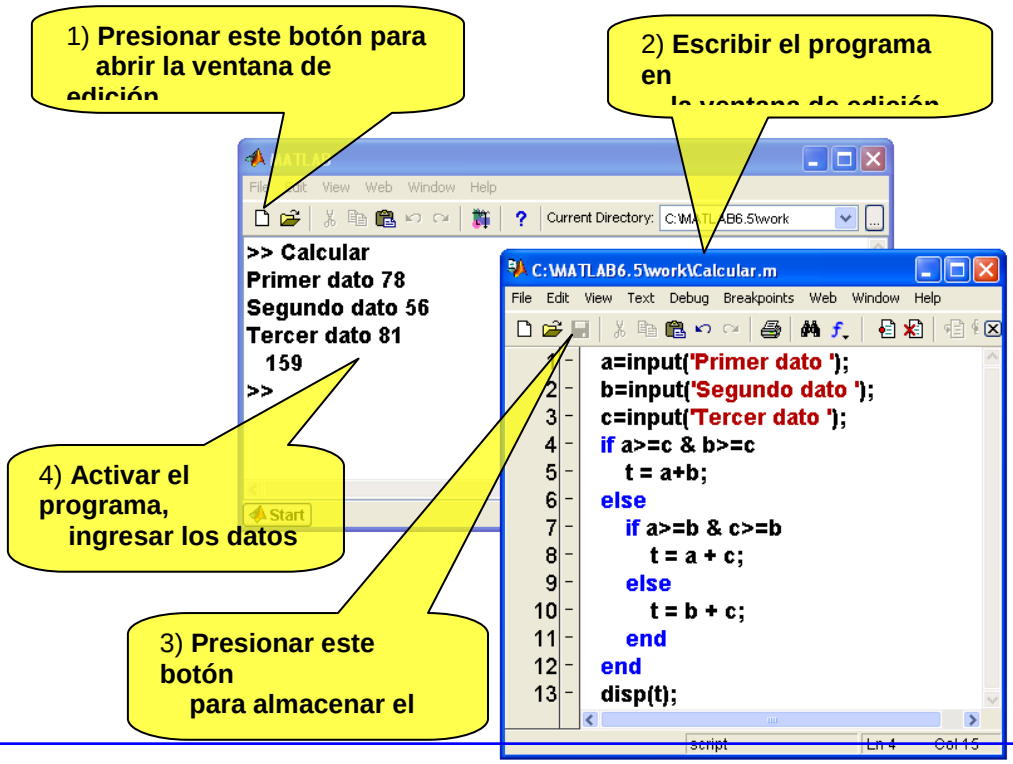
```
>> g=diff(f)
g =  -2000/x^2+4*pi*x
>> x=fzero(char(g),[4,6])
x =
   5.4193

>> x=solve(g)
x =
[                                  5/pi*4^(1/3)*(pi^2)^(1/3)]
[ -5/2/pi*4^(1/3)*(pi^2)^(1/3)+5/2*i*3^(1/2)/pi*4^(1/3)*(pi^2)^(1/3)]
[ -5/2/pi*4^(1/3)*(pi^2)^(1/3)-5/2*i*3^(1/2)/pi*4^(1/3)*(pi^2)^(1/3)]
>> x=eval(x)
x =
   5.4193
  -2.7096 + 4.6932i
  -2.7096 - 4.6932i
```

14 PROGRAMACIÓN EN MATLAB

Para usar el componente programable de MATLAB debe abrir una ventana de edición presionando el botón **New M-File** en la barra de opciones de MATLAB:.

Escribir el programa en la ventana de edición y almacenarlo con algún nombre. Finalmente, activar el programa escribiendo el nombre en la ventana de comandos. Ingresar los datos y obtener los resultados.

Ejemplo. Escribir y probar un programa en MATLAB para obtener la suma de los dos valores mas altos de tres números ingresados como datos.



14.1 Archivos de comandos

Para crear un archivo de comandos (programa, o script), seleccione en la barra de herramientas de MATLAB la opción**: File → New → M-file** o presione en el ícono respectivo. Se abrirá una ventana de edición

En la ventana de edición Escribir los comandos y almacénelos en un archivo con algún nombre. Puede incluir comentarios con el símbolo **%**

Ejemplo. Escribir en la ventana de edición las instrucciones para graficar **sen(x)** entre **0 y 2π**

```
x=0:0.1:2*pi;
y=sin(x);
plot(x,y);
grid on
```

Guarde el archivo con algún nombre, ejemplo: **prueba**
Para ejecutar el programa Escribir en la ventana de comandos

```
>> prueba
```

Para editar un archivo de comandos, seleccione en la barra de herramientas de MATLAB la opción**: File → Open** o seleccione el ícono respectivo.

En el archivo abierto en la ventana de edición haga los cambios y guárdelo nuevamente.

14.2 Estructuras de control de flujo en MATLAB

14.2.1 Instrucciones de Entrada y Salida

Ingreso de un dato desde el teclado: **variable=input('mensaje');**

Ej.

```
x = input('ingrese un dato ');
```

Salida de un resultado a la pantalla: **disp(valor)**

Ej.

```
x=exp(2);
disp(x);
```

Salida de mas de un resultado a pantalla: **disp([valor, valor, …]);**

Ej.

```
x=2^7;
y=sqrt(pi);
disp([x, y]);
```

Salida de resultados formateados a pantalla: **fprintf('formatos',variables)**

Ej.

```
x=2^7;
y=sqrt(pi);
fprintf('%d %f'',x,y);
```

Puede especificar cantidad de columnas y decimales:

Ej.

```
x=2^7;
y=sqrt(pi);
fprintf('%5d %8.3f'',x,y);
```

Otras especificaciones de formato puede verlas con **help fprintf**

14.2.2 Decisiones (instrucción if)

if condición instrucciones **end**	**if** condición instrucciones **else** instrucciones **end**

Ej. Escribir y almacenar el siguiente programa para mostrar el mayor entre dos datos:

```
a=input('ingrese el primer dato ');
b=input('ingrese el segundo dato ');
if a>b
  m=a;
else
  m=b;
end
disp(m);
```

Guárdelo con el nombre **prueba** y úselo desde la ventana de comandos:

```
>> prueba
      ingrese el primer dato 5
      ingrese el segundo dato 8
         8
```

interacción con MATLAB

14.2.3 Decisiones múltiples (instrucción switch)

Ej. Escribir un siguiente programa para instrumentar la definición:

$$y = \begin{cases} 3, & x = 1 \\ 2x+1, & x = 2,3,4 \\ e^x, & \text{otro } x \end{cases}$$

```
x=input('dato ');
switch x
    case 1,
        y=3;
    case {2,3,4},
        y=2*x+1;
    otherwise
        y=exp(x);
end
disp(y);
```

Almacénelo con algún nombre. Ejemplo **prueba2**
Para usarlo escriba en la ventana de comandos

```
>> prueba2
```

14.2.4 Repetición condicionada al inicio (instrucción while)

```
while condición
  instrucciones
end
```

Ej. Sumar los n primeros términos de la serie armónica:

```
n=input('cantidad de terminos ');
s=0;
while n>0
  s=s+1/n;
  n=n-1;
end
disp(s);
```

Almacénelo con algún nombre. Ejemplo **prueba3**
Para usarlo escriba en la ventana de comandos

```
>> prueba3
```

14.2.5 Repetición condicionada a una secuencia (instrucción for)

```
for variable=inicio: incremento: final
  instrucciones
end
```

Ej. Sumar los n primeros términos de la serie armónica:

```
n=input('cantidad de terminos ');
s=0;
for i=1:n
  s=s+1/i;
end
disp(s);
```

Almacénelo con algún nombre. Ejemplo **prueba4**
Para usarlo escriba en la ventana de comandos

>> prueba4

14.2.6 Puede interrumpir una repetición (instrucción break)

Ej. Leer n datos. Calcular y mostrar la raíz cuadrada. Pero si entra un valor negativo, mostrar un mensaje y terminar

```
n=input('cantidad de datos ');
for i=1:n
   x=input('ingrese siguiente dato ');
   if x<0
     disp('Error');
     break;
   else
     r=sqrt(x);
     disp([x,r]);
   end
end
```

Para ver la descripción de las estructuras del lenguaje de MATLAB, escriba

>> help lang

Funciones en MATLAB

En general una función en los lenguajes de programación es un conjunto de instrucciones que se escriben separadamente del programa y que realizan alguna tarea especificada. Los usuarios pueden definir funciones y agregarlas a las funciones propias de MATLAB.

El mecanismo usual para trasmitir datos a las funciones es mediante una lista de variables que se denominan parámetros. Sin embargo, a diferencia de los programas, las variables que se usan dentro de una función, no están disponibles fuera de ella, a menos que se use una declaración explícita y que se verá mas adelante.

Declaración de una función en MATLAB

```
function  variable = nombre (parámetros)
    instrucciones
```

variable contendrá el resultado que entrega la función
parámetros son variables que reciben los datos que entran a la función
nombre identificación de la función
instrucciones se incluyen en la función según la tarea especificada

El nombre asignado a una función debe coincidir con al nombre usado en la declaración de la función.

Las funciones se escriben en la ventana de edición de MATLAB y se las almacena en alguna carpeta. En la ventana de comandos debe especificarse la ruta a esta carpeta.

El uso de una función es similar al uso de las funciones comunes de MATLAB. El nombre debe coincidir con el nombre asignado, aunque los parámetros pueden tener nombres diferentes, pero su uso debe ser coherente.

Ej. Escriba una función para elegir el mayor entre dos números

Abra un documento nuevo en la ventana de edición y escriba:

```
function m = mayor(a, b)
if a>b
  m = a;
else
  m = b;
end
```

m es la variable que entrega el resultado
mayor es el nombre de la función
a, b son los parámetros que ingresan los datos a la función

Almacene esta función con el nombre **mayor**
Suponer que quiere escoger el mayor entre e^{π} y π^{e}.
Escriba en la ventana de comandos:

```
>> a = exp(pi);
>> b = pi^exp(1);
>> m = mayor(a, b)
      23.1407          (respuesta que muestra MATLAB)
```

Los nombres de las variables pueden ser diferentes:

```
>> x = exp(pi);
>> y = pi^exp(1);
>> t = mayor(x, y)
      23.1407          (respuesta que muestra MATLAB)
```

Ej. Escriba una función que reciba un número y determine si es un número primo. El resultado que entrega la función será 1 o 0 según corresponda;

```
function  p = primo( x )
c = 0;
for d = 1: x
  if mod(x, d) == 0
    c = c + 1;
  end
end
if c > 2
  p = 0;
else
  p = 1;
end
```

Guarde la función en el disco con el nombre **primo**
Pruebe la función directamente desde la ventana de comandos

```
>> x = 25;
>> p = primo(x)
      0                (resultado que muestra MATLAB)
>> p = primo(43)
      1                (resultado que muestra MATLAB)
```

Escriba en una nueva ventana de edición un programa que use la función **primo** para encontrar todos los números primos menores a 20:

```
for x = 1: 20
  if primo(x) == 1
    disp(x);
  end
end
```

Almacene su programa en el disco con un nombre **prueba**
En la ventana de comandos pruebe su programa:

```
>> prueba
     1                    (resultados mostrados por MATLAB)
     2
     3
     5
     7
    11
    13
    17
    19
```

Una función puede entregar más de un resultado
Las variables que entregan los resultados deben definirse entre [ ]

Ej. Escriba una función que entregue el área y el volumen de un cilindro dados su radio (r) y su altura (h)

```
function [area, vol] = cilindro(r, h)
area = 2*pi*r*h + 2*pi*r^2;
vol = pi*r^2*h;
```

Escriba y almacene la función con el nombre cilindro.

Use la función para calcular el área y el volumen de una lata de cilíndrica que tiene un diámetro de 10cm y una altura de 12cm

Escriba en la ventana de comandos:

```
>> r = 5;
>> h = 12;
>> [a, v] = cilindro(r,h)
```

MATLAB mostrará los resultados de **a** y **v**

Las variables definidas dentro de una función son locales, es decir que a diferencia de los programas, no son visibles fuera de la función

Ej. Escriba la función:

```
function  x=fn(a, b)
c = a + b;
x = 2*c;
```

Almacenar con el nombre **fn** y usar desde la ventana de comandos:

```
>> a = 3;
>> b = 5;
>> t = fn(a, b)
        t = 16          (resultado que muestra MATLAB)
>> c                    (intentamos conocer el valor de c en la función)
```

??? Undefined function or variable 'c'.

mensaje de error de MATLAB que indica que **c** no está definida

Las variables en los programas son visibles (disponibles) fuera del programa.

Ej. Un programa con acción similar a la función anterior:

```
a = input('ingrese dato ');
b = input('ingrese dato ');
c = a + b;
x = 2*c;
disp(x);
```

Almacene con el nombre prueba y active el programa:

```
>> prueba
        ingrese dato  3        (interacción para ingreso de datos)
        ingrese dato  5
           16                  (resultado que muestra MATLAB)
>> c
        c = 8                  (la variable c puede ser utilizada)
```

Es posible hacer que las variables de una función sean visibles fuera de su ámbito mediante una declaración especial

El comando **global** permite tener acceso a variables de las funciones

Ej. Modifique la función **fn** para que la variable **c** sea visible:

```
function  x=fn(a, b)
global c;
c = a + b;
x = 2*c;
```

Almacene con el nombre fn y use la función:

```
>> global c;
>> a = 3;
>> b = 5;
>> t = fn(a, b)
        t = 16          (resultado que muestra MATLAB)
>> c                    (intentamos conocer el valor c de la función)
        c=8             (la variable c está disponible ahora)
```

Una función puede no necesitar parámetros

Ej. Escriba una función que lea y valide un entero entre 1 y 5

```
function n=entero
x=0;
while x==0
  n=input('ingrese un entero entre 1 y 5 ');
  if n>0 & n<6
    x=1;
  end
end
```

Una función puede no entregar resultados ni usar parámetros

Ej. Escriba una función que muestre un menú en la pantalla

```
  function menú
clc;
  disp('1) ingresar');
  disp('2) borrar');
  disp('3) salir');
```

para usa esta función escriba

```
>> menu
```

Una función puede recibir como parámetros vectores o matrices.

Ej. Escriba una función que reciba un vector y entregue el promedio del valor de sus elementos.

```
function p=prom(x)
n=length(x);
s=0;
for i=1:n
   s=s+x(i);
end
p=s/n;
```

Esta función es equivalente a la función **mean** de MATLAB

Para usar la función creada debe definir el vector antes de llamarla
La longitud del vector se determina con la función **length** de MATLAB

```
>> x=[2 7 3 5 4 7 6];
>> t=prom(x)
```

t = 4.8571 (es el resultado que muestra MATLAB)

Ej. Escriba una función que reciba un vector y entregue la suma de los valores pares únicamente.

```
function s=sumapares(x)
n=length(x);
s=0;
for i=1:n
   if mod(x(i), 2) == 0
      s=s+x(i);
    end
end
```

No existe una función equivalente en MALAB

```
>> x=[2 7 3 5 4 7 6];
>> t=sumapares(x)
```

t = 12 (es el resultado que muestra MATLAB)

Una función puede entregar un vector o matriz

Ej. Escriba una función que entregue un vector de longitud n conteniendo números aleatorios enteros con valor entre 1 y 6:

```
function d=dados(n)
for i=1:n
   d(i)=fix(rand*6+1);
end
```

Para usar esta función debe enviar un valor para el parámetro n:

```
>> t=dados(5)
        t = 6   3   4   3   2
```

(es un vector resultante de MATLAB)

Una solución alternativa para el ejemplo anterior:

```
function d=dados(n)
d=[ ];
while length(d) < n
   d = [d, fix(rand*6+1)]
end
```

Una función puede recibir y entregar vectores o matrices

Ej. Escriba una función que reciba un vector y entregue otro vector conteniendo los elementos cuyo valor es un número par.

```
function y=pares(x)
n=length(x);
y=[ ];
for i=1:n
  if mod(x(i), 2) == 0
     y = [y, x(i)];
   end
end
```

```
>> x=[2 7 3 5 4 7 6];
>> t=pares(x)
    t =
        2  4  6
```

(es el resultado que muestra MATLAB)

Ej. Escriba una función que reciba dos vectores A, B y entregue un tercer vector que contenga los elementos que están en ambos vectores:

```
function C=interseccion(A,B)
n=length(A);
m=length(B);
k=1;
for i=1:n
   for j=1:m
      if A(i) == B(j)
 C(k) = A(i);
         k = k + 1;
      end
   end
end
```

Para usar esta función debe definir los vectores que entran. Recuerde que pueden tener nombres diferentes a los que usa la función:

```
>> A=[2 7 5 4 3 8];
>> B=[7 1 3 9 0];
>> C=interseccion(A,B)
```

C = 7 3 (Es el vector resultante que entrega MATLAB)

Una forma alterna para escribir la función anterior

```
function C=interseccion(A,B)
C=[ ];
for i=1: length(A)
   for j=1: length(B)
      if A(i) == B(j)
 C = [C , A(i)];
      end
   end
end
```

Ej. Defina una función para elegir una muestra aleatoria de tamaño **n** de una población de tamaño **m**

```
function x=muestra(m, n)
x=[ ];
while length(x)<n
   t=fix(rand*m)+1;
   x=[x, t];
end
```

Ej. Defina una función para elegir una muestra aleatoria de tamaño **n** de una población de tamaño **m**. En esta versión. los elementos no deben estar repetidos en la muestra

```
function x=muestrasr(m,n)
x=[ ];
while length(x)<n
   t=fix(rand*m)+1;
   if ismember(t, x) == 0
      x=[x, t];
   end
end
```

Uso de la función muestra

```
>> x=muestrasr(10,4)
x =
    5    10     9     6
>>
```

Ej. Escriba una función para obtener una tabla de n elementos (diferentes) del pozo millonario (números aleatorios entre 1 y 25):

```
function t=pozo(n)
t=[ ];
while length(t)<n
   x=fix(rand*25)+1;
   if ~ismember(x,t)
      t=[t,x];
   end
end
t=sort(t);
```

```
>> t=pozo(8)
t =
         5     8    10    11    13    14    18    21
```

Ej. Escriba un programa para generar **m** tablas diferentes de **n** elementos del pozo millonario (números aleatorios diferentes entre 1 y 25):

```
k=input('¿cuantas tablas ');
n=input('¿longitud de la tabla? ');
g=[ ];
i=0;
while i<k
   t=pozo(n);
   t=sort(t);
   if ~ismember(t, g, 'rows')
      g=[g; t];
      i=i+1;
   end
end
disp(g)
```

```
>> tablas
¿cuantas tablas 5
¿longitud de la tabla? 7

     6    10    12    15    17    18    20
     1     2     8    11    16    20    22
     6     9    11    13    17    20    25
     6     7    11    18    19    24    25
     4     6    10    12    15    16    21
```

Si la salida de una función es antes del final, se debe usar return

Ej. Escriba una función para determinar si los elementos de un vector están en orden creciente:

Si el resultado es 1, los elementos están en orden creciente, caso contrario, el resultado es 0

```
function t=orden(x)
n=length(x);
for i=1:n-1
   if x(i) > x(i+1)
         t=0;
         return;
   end
end
t=1;
```

Un programa puede llamar a funciones

Ej. Una función para eliminar espacios intermedios de una frase:

f: contiene una frase que entra a la función
x contiene la frase sin espacios intermedios

```
function x=compactar(f)
n=length(f);
x='';
for i=1:n
   if f(i) ~= ' '
      x = strcat(x, f(i));
   end
end
```

Una forma alterna de escribir la función anterior

```
function x=compactar(f)
n=length(f);
x='';
for i=1:n
   if f(i) ~= ' '
      x = [x, f(i)];
   end
end
```

Un programa que lee una frase, usa la función **compactar** para eliminar los espacios intermedios, y luego muestra un mensaje en caso de que la frase sea simétrica: sus caracteres opuestos son iguales

```
f=input('ingrese una frase ');
f=compactar(f);
n=length(f);
sim=1;
for i=1:n/2
   if f(i) ~= f(n-i+1)
      sim=0;
   end
end
if sim == 1
   disp('la frase es simetrica');
else
   disp('la frase no es simetrica');
end
```

Probamos este programa suponiendo que lo hemos almacenado con el nombre **prueba**:

>> prueba
ingrese una frase 'anita lava la tina'; (dato que ingresa)

la frase es simetrica (resultado de MATLAB)

Una función puede llamarse a si misma.
Estas funciones se denominan funciones recursivas

Ej. Use la siguiente definición recursiva para calcular el máximo común divisor entre dos números enteros. Escriba una función con esta definición

$$mcd(a,b) = \begin{cases} mcd(a-b,b), & a > b \\ mcd(a,b-a), & b > a \\ a, & a = b \end{cases}$$

```
function c=mcd(a, b)
if a>b
    c=mcd(a-b, b);
else
    if b>a
        c=mcd(a, b-a);
    else
        c=a;
    end
end
```

Use la función:

```
>> x=mcd(36, 48)
```

Una función especial de MATLAB para desplegar mensajes de error y terminar la ejecución: error

Ej.

```
if d<0
   error('valor incorrecto');
end
```

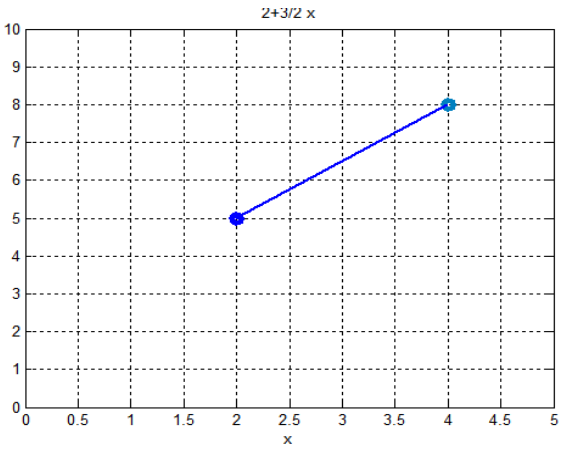
Funciones que entregan resultados analíticos

Ejemplo. Escriba y almacene una función que reciba dos puntos y entregue la ecuación de la recta que incluye a estos dos puntos:

```
function y=recta(x1, y1, x2, y2)
syms x;
m=(y2 - y1)/(x2 - x1);
y=y1 + m*(x - x1);
```

Uso de la función desde la línea de comandos y su gráfico

```
>> x1=2;
>> y1=5;
>> x2=4;
>> y2=8;
>> y=recta(x1, y1, x2, y2)
      y =
          2+3/2*x
>> plot(x1,y1,'o',x2,y2,'o')
>> hold on
>> ezplot(y, [2,4])
>> grid on
>> axis([0,5,0,10])
>>
```



15 INTERACCIÓN DE MATLAB CON OTROS ENTORNOS

Interacción con EXCEL

1) Importar una tabla de datos desde Excel a una matriz en MATLAB

a) En **Excel** cree la tabla y almacénela con formato tipo texto delimitado con tabulaciones. Elija algún nombre. Ejemplo **T.txt**

b) En **MATLAB** cargue la tabla **T** y úsela como una matriz:

```
>> load  T.txt;
>> A=T
```

2) Exportar una matriz de datos desde MATLAB a una tabla en Excel

a) En **MATLAB** cree una matriz y almacénela con el comando **save** con el siguiente formato. Elija los nombres. Ejemplo

A: nombre de la matriz en MATLAB
T: nombre para la tabla almacenada

```
>> save  T   A  -ascii
```

b) En **Excel** abra el archivo **T** y úselo como una tabla de datos

16 BIBLIOGRAFÍA

The MathWorks, Inc. *Using MATLAB Computation, Visualization, Programming*, version 6