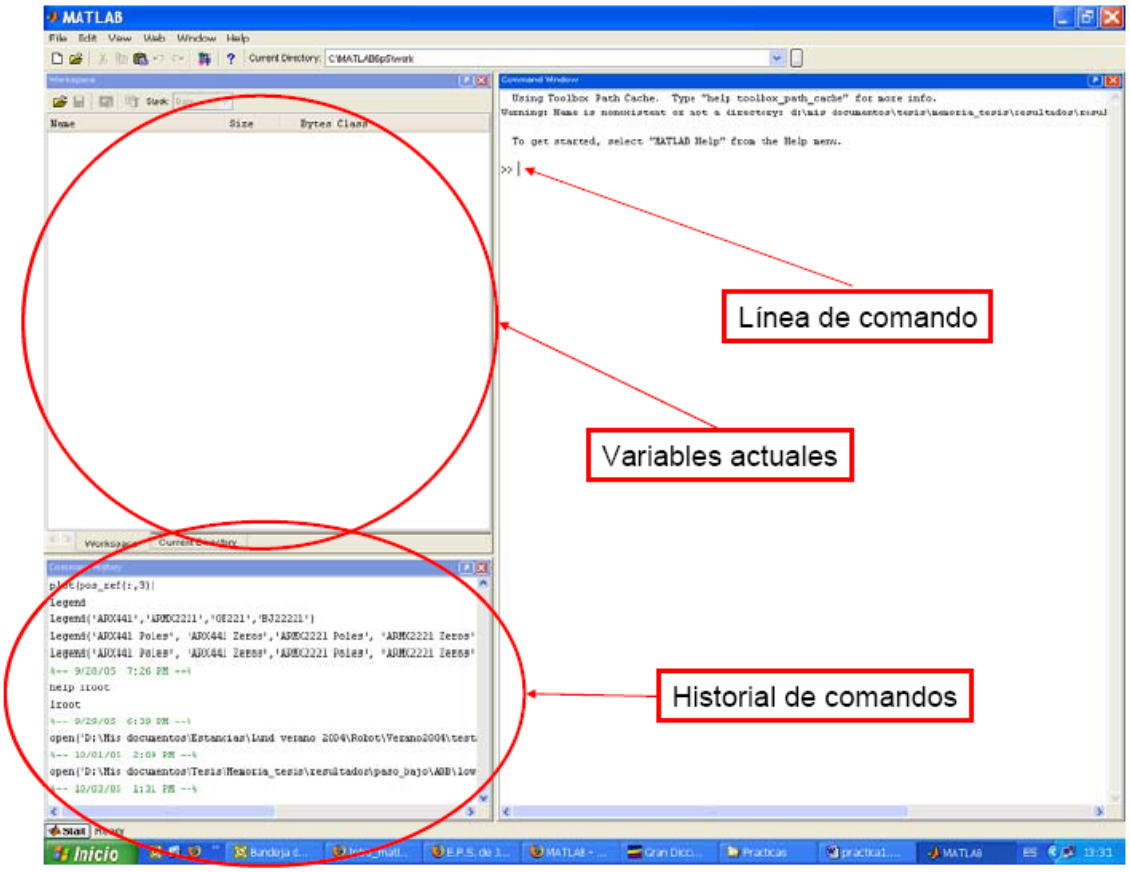
	Curso	**Métodos Numéricos**	**Código : MB536**
	Tema	**Introducción al MATLAB**	
	Practica	**01**	
	Profesores	**Garrido Juárez, Rosa** **Hermes Pantoja, Carhuavilca** **Ruiz Lizama, Edgar**	**Castro Salguero, Robert** **Obregón Ramos, Máximo**

1. Objetivos :

Manejar los comandos básicos del MATLAB para operaciones con escalares, vectores y matrices.

2. Introducción MATLAB como una Calculadora



2.1 ¿Cómo se escriben los números?

- Enteros (sin punto decimal): **23 321 -34**
- Reales (con punto decimal): **23. -10.1 11.321**
- Reales (notación científica o exponencial): **2.e-2=2x10^{-2}=0.02**

2.2 MATLAB como una Calculadora

Los operadores básicos son: + - * / ^

```
>> 2 + 3/4 * 5+4^-1+(1/5)^(-3/2)
>> 2 + 3/(4*5) +4^-0.5+(1/5)^(-3/2)
```

Se puede utilizar MATLAB como simple calculadora, escribiendo expresiones aritméticas y terminando con **RETURN**. Se obtiene el resultado inmediatamente a través de la

variable del sistema **ans** (answer). Si no se desea eco (es decir, la respuesta inmediata a cada orden) al final de cada instrucción, debe finalizarse con "**punto y coma**".

MATLAB trabaja de acuerdo a las prioridades:

1. Expresiones entre paréntesis
2. Potencias 2+3^2$\Rightarrow$ 2 + 9
3. * , / trabajan de izquierda a derecha (3*4/5= 12/5)
4. + , - trabajan de izquierda a derecha (3+4-5= 7-5)

2.3 Ejercicios

Escribir en MATLAB las siguientes expresiones matemáticas

a) $\dfrac{3+e^2}{\dfrac{2}{\sqrt[7]{3}}-\left(\dfrac{1}{3.1-2}\right)^{3/\pi}}$ b) $\dfrac{1}{\dfrac{2}{0.1^{1/2}}-\dfrac{\arctan(0.17)}{2^{1/3}}+a\cos(0.1)}$

c) $\dfrac{1}{\dfrac{5}{0.1^{1/\pi}}-\dfrac{asen(0.17)}{2^{e/3}}}$ d) $\dfrac{1+\sqrt{3\cdot e^{3+\sqrt{2}}}}{\sqrt{e+1}\cdot\pi}\cdot\sqrt[\pi]{e+\pi}$

3. Números y Formatos

MATLAB reorganiza diferentes clases de números

Tipo	Ejemplo
Entero	1362, -217897
Real	1.234, -10.76
Complejo	3.21 – 4.3i (i = √-1)
Inf	Infinito(Resultado de dividir entre 0)
NaN	No es un número (0/0)

La notación "e" es usada para números muy grandes o muy pequeños:
-1.3412e+03 = - 1.3412×10^{3} = -1341.2
-1.3412e-01 = - 1.3412×10^{-1} = -0.13412

4. Operadores relacionales:

- < Menor (para complejos sólo afecta a partes reales)
- <= Menor o igual (sólo afecta a partes reales)
- > Mayor (sólo afecta a partes reales)
- >= Mayor o igual (sólo afecta a partes reales)
- x = = y Igualdad (afecta a los números complejos)
- x ~ = y Desigualdad (afecta a los números complejos)

5. Formatos numéricos

Comando	Resultado
format short	1.3333 0.0000
format short e	1.3333e+000 1.2345e-006
format long	1.33333333333333 0.00000123449932
format long e	1.333333333333333e+000 1.234499317939127e-006
format rat	1/3

6. Funciones matemáticas elementales

sqrt(x)	raíz cuadrada	sin(x)	seno(en radianes)
exp(x)	exponencial	cos(x)	coseno(en radianes)
log(x)	logaritmo neperiano	tan(x)	tangente(en radianes)
log10(x)	logaritmo decimal	asin(x)	arco seno
abs(x)	valor absoluto	acos(x)	arco coseno
		atan(x)	arco tangente

El argumento de las funciones puede ser un número, una variable o una expresión. Cuando en una expresión aparece alguna función, su valor se calcula antes que cualquier otra cosa.

Ejemplos

```
>> sqrt(7)
>> sqrt(7/5)
>> a=2.1; sqrt(2*a)
>> exp(3)
>> 7*exp(5/4)+3.54
>> x = 5*cos(pi/6) , y=5*sin(pi/6)
>> acos(x/5), asin(y/5)
```

Ejercicio: Evaluar las siguientes expresiones matemáticas en MATLAB.

a) $\text{Log}_{9.8}(2)$ c. $\sqrt{5} + e^2$

b) | Arcsen(-0.5) | d. tan(e)

7. Otras funciones elementales

Otras funciones matemáticas elementales	Comentarios
angle(x)	Angulo de fase de un valor complejo x
real(x)	Parte real de una valor complejo x
imag(x)	Parte imaginaria de un valor complejo x
conj(x)	Conjugada de un complejo x
round(x)	Redondea al entero más próximo
fix(x)	Redondea un valor real hacia 0
floor(x)	Redondea hacia $-\infty$
ceil(x)	Redondea hacia $+\infty$
sign(x)	+1 si x>0 y -1 si x<0
rem(x,y)	Residuo al dividir dos enteros x e y

8. Vectores y Matrices

Selección de los elementos de un vector x o matriz A

x(n)	Devuelve el n-ésimo elemento del vector x
x([n,m,p])	Devuelve los elementos del vector x situados en las posiciones n-ésimas,m-ésimas y p-ésimas
x(n:m)	Devuelve los elementos del vector x situados entre el n-ésimo y el m-ésimo,ambos inclusive.
x(n:p:m)	Devuelve los elementos del vector x situados entre el n-ésimo y el m-ésimo, ambos inclusive pero separados de p en p unidades.
A(m,n)	Devuelve el elemento (m,n) de la matriz A (fila m y columna n)

A([m,n],[p,q])	Devuelve la submatriz de A formada por la intersección de las filas n-ésima y m-ésima y las columnas p-ésima y q-ésima.
A(n,:)	Devuelve la fila n-ésima de la matriz A
A(:,p)	Devuelve la columna p-ésima de la matriz A
A(:)	Devuelve el vector columna cuyos elementos son las columnas de A situadas por orden
A(:,:)	Devuelve toda la matriz A

Funciones matriciales

Función	Descripción
rank(A)	Rango de la matriz A.
det(A)	Determinante de la matriz A
trace(A)	Traza de la matriz A, suma de los elementos de la diagonal
inv(A)	Calcula la inversa de la matriz A
A'	Transpuesta de la matriz A.
diag(A)	Extrae la diagonal de la matriz A como vector columna
eye(n)	Crea la matriz identidad de orden n
eye(m,n)	Crea la matriz de orden mxn con unos en la diagonal principal y ceros en el resto.
zeros(m,n)	Crea la matriz nula de orden mxn
ones(m,n)	Crea la matriz de orden mxn con todos sus elementos unos
[n,m]=size(A)	Devuelve el número de filas y columnas de la matriz A

9. Operaciones de matrices

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

- [A,B,C] Devuelve la matriz formada por las submatrices A,B,C,

10. Representación gráfica de funciones

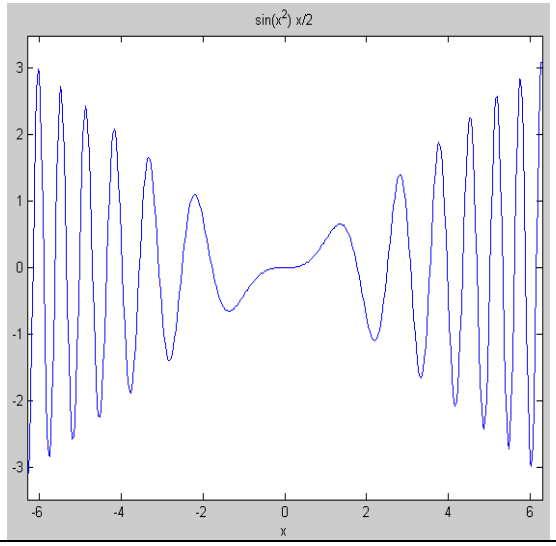
La forma más sencilla de dibujar con MATLAB una función y=f(x) es:

>> ezplot('expresion de la funcion')

Esta orden dibuja la gráfica de la función dada por la expresión, para x variando en en un intervalo por defecto.

>> ezplot('sin(x^2)*x/2',[-2*pi 2*pi])
dibuja la gráfica de la función

$$f(x) = \frac{x\sin(x^2)}{2}, \quad x \in [-2\pi, 2\pi]$$



Si se desea dibujar la función en un intervalo distinto, [a,b], hay que indicarlo expresamente en la orden:

>> ezplot('expresion de la funcion',[a,b])

También se pueden dibujar varias gráficas separadas en la misma ventana, usando la orden

>> subplot(m,n,p)

Este comando permite dividir la ventana gráfica en una matriz mxn de sub-ventanas gráficas, activando para dibujar la p-ésima de ellas.

plot(y) : produce un gráfico lineal de los elementos del vector y versus los índices de y.

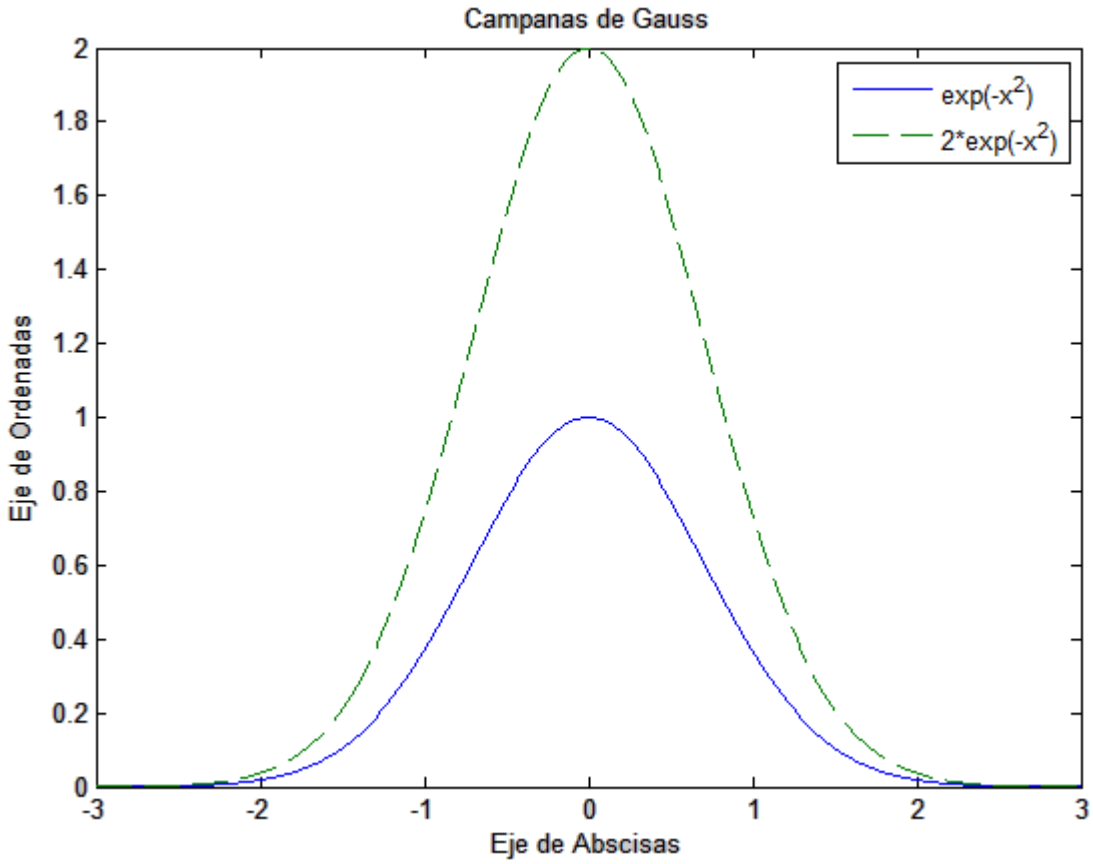
```
>>y = [ 0 0.6 0.9 0.1 0.8 0.3 0.4];
>>plot(y)
```

plot(x,y) : Dados dos vectores de la misma dimensión, x e y, produce un gráfico lineal de los elementos del vector x versus los elementos del vector y

```
>>t = 0 : 0.1 : 4*pi;
>>y = sin(t);
>>plot(t,y,'r')
```

Otra forma :

```
>>x=linspace(-3,3,500);y=exp(-x.^2);z=2*exp(-x.^2);
>>plot(x,y,'-',x,z,'--')                    % dibujamos dos funciones
>>title('Campanas de Gauss')
>>xlabel('Eje de Abscisas')                 % Etiqueta el eje horizontal
>>ylabel('Eje de Ordenadas')                % Etiqueta el eje vertical
>>legend('exp(-x^2)', '2*exp(-x^2)')        % Leyenda
```



11. Práctica de comandos

1) Crear los siguientes vectores:

$x = \left[0 \ \sqrt{3} \ \pi \ e^2\right]$

$y = \left[0 \quad 0.1\pi \quad 0.2\pi \quad 0.3\pi \quad 0.4\pi \quad 0.5\pi \quad 0.6\pi \quad 0.7\pi \quad 0.8\pi \quad 0.9\pi \quad \pi\right]$

2) Crear un vector **z** de cuatro números complejos
3) Listar el tercer elemento del vector **z**
4) Listar los 5 primeros elementos del vector *y*
5) Listar los 5 últimos elementos del vector *y*
6) Listar los elementos de posiciones impares del vector *y*
7) Listar los elementos de posiciones 2, 4, 5, y 7 del vector *y*
8) Crear los vectores a = [1 2 3 4 5] y b = [1 3 5 7 9]
9) Fusionar los vectores a y b en un vector *c*
10) Obtener la transpuesta del vector *c*

11) Crear las siguientes matrices:.

$$g = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

$$h = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

12) Multiplicar las matrices g y h
13) Multiplicar g con la transpuesta de h
14) Multiplique g y h componente a componente
15) Eleve 2 a cada elemento de g
16) Resolver el sistema:

2 a + 3 b + c = 6
4 a + b + 2 c = 7
6 a + b + 7 c = 4

Mediante la función **inv**.

17) Resuelva el sistema anterior mediante **\.**
18) Utilizando MATLAB determine el valor de la expresión

$$\left[\frac{(9.8-0.8)}{4-\ln(2)}\right]^{\frac{245}{124}} = ?$$

19) Crear la siguiente matriz

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 0 & 0 & 5 & 6 & 7 & 8 \\ 0 & 0 & 3 & 0 & 9 & 10 & 11 & 12 \\ 0 & 0 & 0 & 4 & 20 & 0 & 5 & 4 \\ 1 & 5 & 9 & 20 & 0 & 0 & 0 & 0 \\ 2 & 6 & 10 & 0 & 0 & 0 & 0 & 0 \\ 3 & 7 & 11 & 5 & 0 & 0 & 0 & 0 \\ 4 & 8 & 12 & 4 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Luego extraer la siguiente submatriz

$$T1 = \begin{bmatrix} 3 & 0 & 9 & 10 \\ 0 & 4 & 20 & 0 \\ 9 & 20 & 0 & 0 \\ 10 & 0 & 0 & 0 \end{bmatrix}$$

20) Escriba las matrices A y B definidas por

$$A(i,j) = 10(i-j)+1; \quad i,j = 1,\ldots,10$$

$$B(i,j) = \begin{cases} 1, & i-j=1 \\ 0, & \text{en otro caso} \end{cases} \qquad i,j = 1...20$$

21) Obtener la siguiente Matriz:

$$R=\begin{bmatrix} 1 & -2 & 0 & 0 & 0 \\ -2 & 1 & -2 & 0 & 0 \\ 0 & -2 & 1 & -2 & 0 \\ 0 & 0 & -2 & 1 & -2 \\ 0 & 0 & 0 & -2 & 1 \end{bmatrix}$$

Sug: Use la función **diag**.

22) Si b = [1,2,3,4,5]', resuelva el sistema Rx=b

23) Crear la siguiente matriz, dado el orden de la matriz "n".

$$D=\begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 & -2 & 1 \\ 1 & 0 & 0 & \dots & 0 & 1 & -2 \end{bmatrix}$$

24) Graficar la siguiente función

$$f(x)=\begin{cases} x^2 & \textbf{si} \quad x<0 \\ 1 & \textbf{si} \quad 0\le x\le 1 \\ -x+2 & \textbf{si} \quad 1\le x \end{cases}$$

25) Graficar la función

$$y=e^{-x^3} \qquad x\in[0,2\pi]$$

26) Definir las siguientes matrices en MATLAB:

$$A=\begin{pmatrix} 1 & 3 & 5 & 8 \\ 2 & 6 & 5 & 3 \\ 4 & 1 & 9 & 7 \\ 1 & 8 & 0 & 2 \end{pmatrix}; B=\begin{pmatrix} 1 & 9 & 5 & 8 \\ 12 & 5 & 5 & 9 \\ 4 & 2 & 9 & 74 \\ 0 & 6 & 0 & 3 \end{pmatrix}; C=\begin{pmatrix} 1 & 9 \\ 10 & 2 \end{pmatrix}$$

- Realizar los siguientes cálculos básicos con estas matrices:
  $3\cdot A$, $A-7$, $A\cdot B^{T}$, A^{-1}, B^{-1}
- Realizar ahora los siguientes cálculos, siendo D la submatriz de A formada por las 1ª y 3ª filas y columnas, y E la submatriz de B formada por las 2ª y 4ª filas y columnas:
  $D\cdot E^{T}$, $D\cdot C$, $C\cdot E$
- Resolver las siguientes ecuaciones:
  $A\cdot x=B$, $D\cdot x=C$
- Siendo F la submatriz de A formada por las filas 2, 3 y 4, y G la submatriz de B formada por las columnas 1, 2 y 4, calcular: $F\cdot G$

	Curso	**Métodos Numéricos**	**Código : MB536**
	Tema	**Programación en MATLAB y Teoría de Errores**	
	Practica	**02**	
	Profesores	**Garrido Juárez, Rosa** **Castro Salguero, Robert** **Pantoja Carhuavilca, Hermes** **Obregón Ramos, Máximo** **Ruiz Lizama Edgar**	

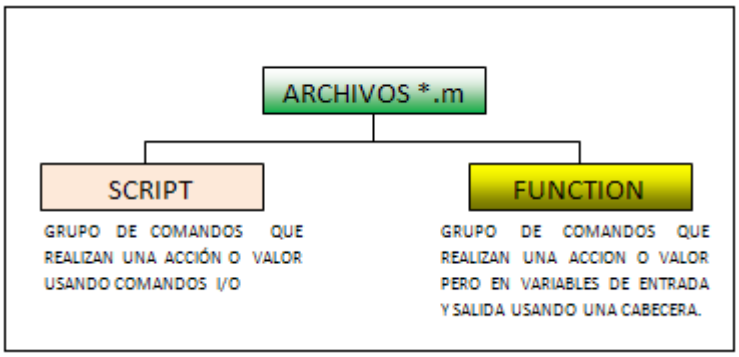
1.Objetivos

Aplicar las instrucciones de control en MATLAB, para la implementación de programas, también estudiar del comportamiento de los errores y la notación en punto flotante.

2. Fundamento Teórico

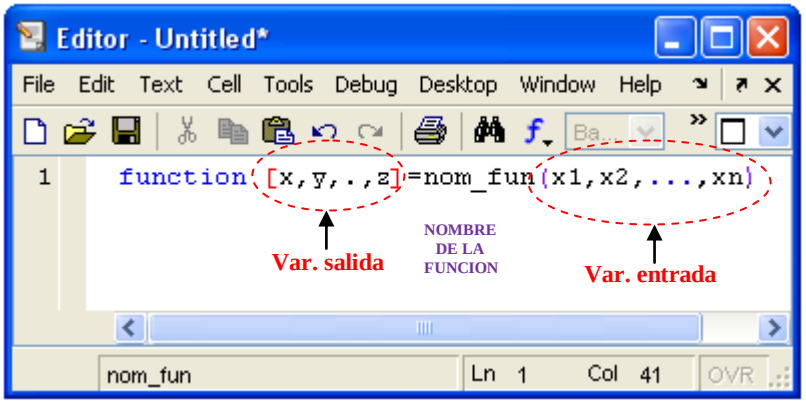
2.1 Archivos *.m

Estos son de dos clases: Script y Function



Estos archivos se editan en la ventana de edición: File/New/M_file

Cabecera de una función:



[Variables de salida] : si es más de un variable se separan por comas.
si es solo una se puede omitir los [ ]

(Variables de entrada) : si es más de una separadas por comas.

Las funciones (functions) se deben grabar como archivo m (M-file) y el nombre del archivo debe ser igual al nombre de la función.

Instrucciones básicas en MATLAB

Controles de Flujo y Sentencia de Decisión

La sentencia IF

Condicional simple	Condicional doble	Condicional múltiple
if <condición> **sentencias;** **end**	**if <condición>** **sentencias1** **else** **sentencias2** **end**	**if** <condicion1> sentencias1 **elseif** <condicion2> sentencias2 **elseif** <condicion3> sentencias3 **else** sentenciasN **end**

La sentencia SWITCH

```
switch selector
   case valor1
      sentencias1
   case valor2
      sentencias2
      .....
   otherwise
      sentenciasN
 end
```

Operadores lógicos y relacionales

Operadores relacionales: **< , > , <=, >=, == (igual), ~= (distinto).**
Operadores lógicos : & (y), | (o), ~ (negación).

Las sentencias FOR y WHILE

Sentencia FOR(Para-Desde)	Sentencias WHILE(Mientras)
for contador=vector Sentencias **End**	**while** condicion Sentencias **end**

La sentencias BREAK

La sentencia **break** hace que se termine la ejecución del bucle mas interno de los que comprenden a dicha sentencia.

Entrada y Salida en un archivo script

Salida:

disp.....Visualiza texto en pantalla (salida)

ejemplo: **disp**('hola')

error. . . . . Visualiza texto en caso de error

ejemplo: `error('no se puede ejecutar')` termina el archivo .m.

fprintf..... Escribe texto con formato

ejemplo: var1=555; **fprintf**('el resultado es %3i',var1)
var2=3.7; **fprintf**('el resultado es %3.1f\n',var2)
var3='hola'; **fprintf**('el resultado es %s\n',var3)
var4='X'; **fprintf**('el resultado es %c\n',var4)
fprintf(' '%s el valor de la variable %c es %3i y %3.1f\n','',var3,var4,var1,var2)

Entrada:

Input: Permite la entrada de valores desde el teclado y se asigna en variables

2.3 Teoría de Errores

Tipos de errores:

a. Errores de redondeo
b. Errores de truncamiento o aproximación

Definición de errores

Si ***a*** es una aproximación a ***A***, entonces se define el **error absoluto** como $\xi_a = |A-a|$, y el **error relativo** como $\delta_a = \frac{\xi_a}{|A|}$ siempre que **A** no sea cero.

Propagación de Errores

a) Funciones de una variable: $y = f(x)$

$$\xi_y \approx \left|\frac{\partial f}{\partial x}\right| \xi_x$$

b) Funciones de varias variables: $y = f(x_1, x_2, \cdots, x_n)$

$$\xi_y \approx \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}\right| \xi_{x_i}$$

2.4 Errores de redondeo y aritmética de punto flotante

Un número flotante (normalizado, en base 2) con precisión **t** se expresa así:

$$\boxed{fl(x) = \pm(1.b_1 b_2 \ldots b_{t-1})_2 \times 2^E}$$

En el sistema simple precisión (t=24), se tiene $fl(x) = \pm(1.b_1 b_2 \ldots b_{23})_2 \times 2^E$

Si el número **1** se expresa $(1.00\ldots0)_2$, entonces el primer número mayor es el que se obtiene sumándole 1 (un) bit, es decir $(1.00\ldots1)_2$

Expresando este valor mediante las potencias (negativas) de 2, se obtiene:

$1\times 2^0 + 0\times 2^{-1} + 0\times 2^{-2} + \ldots + 1\times 2^{-(t-1)} = 1 + 2^{-(t-1)}$

Al reemplazar t=53 y luego efectuar la diferencia de **1** se obtiene el "épsilon de maquina" para este sistema de "doble precisión"

$\varepsilon = (1+2^{-52}) - 1 = 2^{-52}$

Se llama épsilon (ε) de máquina al número (gap) que existe como diferencia entre el 1 y el número próximo más grande.

- *Punto flotante de precisión* **(**eps es la precisión de la maquina: si 1 < x < 1 + eps, entonces x = 1 **)**
- *Punto flotante underflow* **(**x_min es el cero de la maquina: si 0 <= x < x_min, entonces x = 0 **)**
- *Punto flotante overflow* **(**x_max es el infinito de la maquina: si x > x_max, entonces x = inf **)**

3. Parte práctica

Ejemplo 1

Creación de un Archivo Script

a) Crear una carpeta de trabajo en su disquete o en el disco C, usando el **explorador de Windows**, por ejemplo denomínela **MB535_perez.**

b) Establecer la ruta donde el MATLAB buscara su programa, para ello digite en la ventana de comandos la instrucción **cd** seguida de la ruta de su carpeta de trabajo:

>> **cd c:\MB535_perez** y presione la tecla **Enter**

c) Crear un nuevo archivo-m:

Haga clic en el menú **File**, seleccione la opción **New** y haga clic en **M-File**. Aparecerá una ventana en blanco donde deberá digitar su programa.

d) Digite el siguiente programa:

```
% prueba01.m
n=input('Ingrese numero de
periodos=')
x=0:pi/100:2*pi*n;
y=exp(-x/10).*sin(2*x);
plot(x,y)
title('Amortiguamiento')
xlabel('Tiempo(seg)')
ylabel('Posicion (m)')
grid
```

e) Grabar el programa:

Hacer clic en el menú **File**, clic en la opción **Save**, luego digite el nombre del programa: **prueba01** y haga clic en el botón **guardar**.

f) Ejecución del programa:

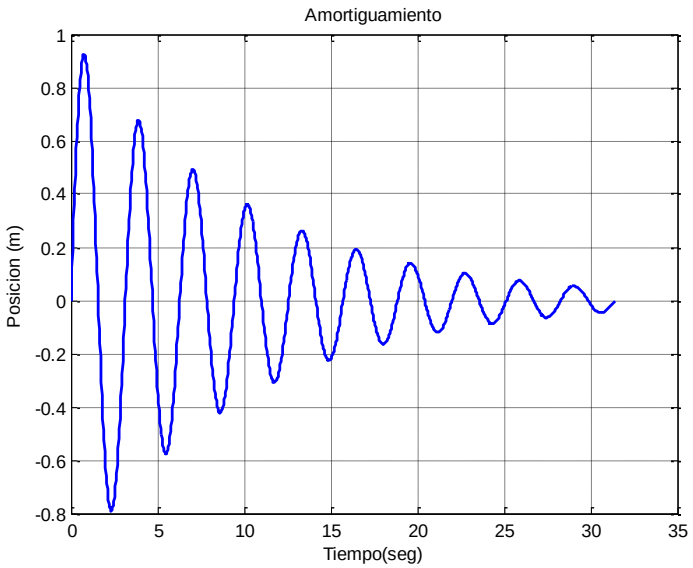
En la ventana de comandos escriba el nombre del programa: **prueba01** y presione la tecla **Enter**.

El programa solicitara el ingreso de un dato:

Ingrese número de períodos =

Digite **5** y presione **Enter**.

Se mostrara el siguiente gráfico:



g) Si el programa no corre correctamente se debe hacer las modificaciones correspondientes, luego volverlo a grabar y ejecutar.

Ejemplo 2

Creación de una función.

Por ejemplo, escriba una función que calcule para un número entero "n", la suma de cifras del Número n y el número de cifras del Número n:

a) Digitar el siguiente código:

```
% cifras.m
% Funcion que calcula para un numero entero n :
% sumcif : Suma de cifras del Numero n
% numcif : Numero de cifras del Numero n
function [sumcif,numcif]=cifras(n)
    sumcif=0;
    numcif=0;
    while n~=0
       digito=rem(n,10);
       sumcif=sumcif + digito;
       numcif=numcif + 1;
       n=fix(n/10);
     end
```

b) Grabarlo con el nombre "**cifras.m**":

c) Llamado de la función:

```
» [sumcif,numcif]=cifras(23400)
sumcif =
    9
numcif =     5
```

Ejemplo 3
Crear una función **expo1** que permita obtener la suma de términos de la serie de Taylor para aproximar el exponencial de un número real x dado n entero:

$$s = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \frac{x^n}{n!}$$

```
% expo1.m
function s=expo1(x,n)
s=1;
for i=1:n
   s=s+x^i/factorial(i);
end
```

Para ejecutarla escriba:
» s=expo1(1,6)
s =
  2.7181

Una variante de esta función puede ser retornando además el error comparado con la función **exp** propia del MATLAB.

```
function [s,err]=expo2(x,n)
% expo2.m
s=1;
for i=1:n
   s=s+x^i/factorial(i);
end
err=abs(exp(x)-s);
```

Para ejecutarla escriba:

» [sum,err]=expo2(1,6)

sum =
   2.7181

err =
  2.2627e-004

Nota.- Obsérvese que el nombre de archivo es idéntico al nombre de la función.

También se pueden declarar funciones en línea:

f=inline('expresion_variables_x_1_x_2_..','x_1','x_2',..)

f : es una variable de memoria.

por ejemplo:

» f=inline('x^2+y^2','x','y')

```
f =
    Inline function:
    f(x,y) = x^2+y^2
» f(3,4)
ans =
   25
```

Ejemplo 4

Funciones recursivas

MATLAB permite la creación de funciones que se llamen a si mismas en tiempo de ejecución para crear algoritmos potentes.

```
% fact.m
function f=fact(n)
if n==0
  f=1;
elseif n==1
  f=1;
else
  f=n*fact(n-1);
end
```

Cuyo llamado se realiza ya sea desde la ventana de comandos o desde otro programa o función que lo requiera:

```
»f=fact(5)
f =
  120
```

Ejemplo 5 IF

```
% prueba02.m
t = rand(1)
if t > 0.75
        s = 0
elseif t < 0.25
        s = 1
else
        s = 1-2*(t-0.25)
end
```

Ejemplo 6 SWITCH

```
% prueba03.m
opc=3
switch opc
case 3
   disp('Mecanica')
case 4
   disp('Mecanica-Electrica')
case 5
   disp('Naval')
case 6
   disp('Mecatronica')
otherwise
```

```
   disp('Fuera de Rango...')
end
```

Ejemplo 7 FOR

```
%prueba04.m
for k=1:100                                  % contador
  x=sqrt(k);                                 % obtiene la raíz de k
  if x>5,                                     % si raíz es mayor a 5
    fprintf('x= %5.2f , k= %3d \n',x,k) % muestra en pantalla x y k
    break                                    % sale del lazo
  end                                        % fin del if
end                                          % fin del for
```

Ejemplo 8 WHILE

```
% prueba05.m
m = 10;
k = 0;
while k<=m
   x = k/10;
   disp([x, x^2, x^3]);   % imprimirá una tabla de valores
   k = k+1;
end
```

Ejemplo 9

Sabiendo que las coordenadas cartesianas de una circunferencia son de la forma x=r*cos(θ) , y=r*sen(θ) crear una **función** que se llame circunferencia1.m que dibuje una circunferencia y que tenga como parámetros de entrada el radio y el ángulo. La función tiene que tener como parámetros de salida todos los pares de valores x,y . Para realizar el programa, hay que tener en cuenta que el radio permanece constante y lo que va cambiando es el ángulo θ.

Solución:

```
function [x,y]=circunferencia1(radio, paso)
    r=radio;
    fi=0;
    i=0;
    if paso>60, error('Angulo muy grande'), end
    for fi=0:paso:360
            fi2=fi*2*pi/360;
            i=i+1;
            x(i)=r*cos(fi2);
            y(i)=r*sin(fi2);
    end
    plot(x,y,'o')
```

Ejemplo 10

Con cuantas cifras significativas aproxima 0.333 a 1/3 ?

Solución:

El número 0.333 coincide con 1/3 en dos cifras significativas, ya que

$$5\times10^{-2} < \frac{\left|\frac{1}{3} - 0.333\right|}{\left|\frac{1}{3}\right|} = 0.001 < 5\times10^{-3}$$

Ejemplo 11

Encuentre la propagación de errores de la siguiente fórmula: $H = Ae\sigma T^4$

con: $\sigma = 5.67 \times 10^{-8}$, $A = 0.1$, $e = 1.0$, y $T = 600^o \pm 20^o$.

(Problema 4.10 del Chapra & Canale, Pág. 103)

Solución

$H = Ae\sigma T^4$

$\sigma = 5.67 \times 10^{-8}$, $A = 0.1$, $e = 1.0$, y $T = 600^o \pm 20^o$.

Error aproximado:

$\varepsilon_H = |\Delta H(T)| = |\partial H/\partial T|\,|\Delta T|$.

Aqui, $\partial H/\partial T = 4Ae\sigma T^3 = 4(0.1)(1.0)(5.67 \times 10^{-8})(600)^3 = 4.90$.

Por lo tanto, $\Delta H(600) = (4.90)(20) = \underline{97.98}$.

Error exacto:

$H_{min} = (0.1)(1.0)(5.67 \times 10^{-8})(580)^4 = 641.65$

$H_{max} = (0.1)(1.0)(5.67 \times 10^{-8})(620)^4 = 837.82$.

Por lo tanto, $\varepsilon_{H_{exact}} = |\Delta H_{exact}| = |(H_{max} - H_{min})/2| = (837.82 - 641.65)/2 = \underline{98.08}$

Este valor es muy cercano al resultado aproximado.

Ejemplo 12

El ensayo de dureza Brinell involucra la compresión de una bola de acero de carburo de tungsteno, de un diámetro D exactamente de 10 mm, contra una superficie, con una carga P en Newtons de 500 ± 1 %, si d es 5.75 mm. medido con una precisión de 0.001 mm, d es el diámetro de la huella impresa en la superficie del material ha ensayar entonces, el número de dureza Brinell HB será:

$$HB = \frac{2P}{\pi D\left(D - \sqrt{D^2 - d^2}\right)}$$

Considere que $\pi = 3.14$ tiene sus 2 cifras decimales exactas.

a) Aproxime HB.
b) Estime el error absoluto de la aproximación HB
c) Estime el rango para el valor exacto de la dureza HB.

Solución

$D = 10$

$P = 500 \quad \xi_P = 5 \quad d = 5.75 \quad \xi_d = 0.001 \quad \pi = 3.14 \quad \xi_\pi = 0.5x10^{-2}$

$$HB = \frac{2P}{\pi D\left(D - \sqrt{D^2 - d^2}\right)} = 17.5132$$

$$\frac{\partial HB}{\partial P} = \frac{2}{\pi D\left(D - \sqrt{D^2 - d^2}\right)} = 0.0350$$

$$\frac{\partial HB}{\partial \pi} = \frac{-2P}{\pi^2 D\left(D - \sqrt{D^2 - d^2}\right)} = -5.5774$$

$$\frac{\partial HB}{\partial d} = \frac{-2PD}{\pi D\left(D - \sqrt{D^2 - d^2}\right)^2 \sqrt{D^2 - d^2}} = -6.7685$$

$$\xi_{HB} = \left|\frac{\partial HB}{\partial P}\right|\xi_P = \left|\frac{\partial HB}{\partial \pi}\right|\xi_\pi + \left|\frac{\partial HB}{\partial d}\right|\xi_d = 0.2098$$

$$HB = 17.5132 \pm 0.2098$$

$$17.3034 \le HB \le 17.7230$$

Ejemplo 13

Una computadora hipotética decimal almacena 6 dígitos significativos (decimal) más 3 dígitos de exponente, y normaliza de modo que el dígito extremo izquierdo sea por lo menos 1. Por esta razón los números representados pueden ser escritos como $\pm(0.dppppp)10^{\pm ppp}$ donde $1 < d < 9$ y $0 < p < 9$. ¿Cuál es el épsilon de la máquina?

Solución:

En esta máquina UNO = +0.100000*E+001 y el número más pequeño +0.000001*E+001; esto es el epsilón, 10^{-5} que sumado a UNO da el siguiente número. O usando la fórmula $\varepsilon = b^{1-t} = 10^{-5}$. (t= precisión , numero de dígitos en la mantisa)

Ejemplo 14

En la siguiente máquina hipotética,

	$e_1e_2e_3$	$m_1m_2m_3$

Escriba en binario:

a) Número positivo más grande normalizado
b) Infinito
c) Uno
d) NaN
e) Número más pequeño normalizado
f) Caracteriza al sistema
g) ¿Cuantos números tendrá este sistema?

Solución

Ei=Ee+Bias, Ei: exponente interno, Ee: exponente externo.
Como se obtiene el Bias : 2^{k-1}-1 , con k= No de bits de Ei
Para esta máquina k=3
Bias=3
Precisión (t)=3 (longitud de la mantisa (M), parte fraccionaria)

a) Número positivo más grande normalizado: Ei=6, M con todos los bits llenos

0	1	1	0	1	1	1

$= +(1.111)2^{6-Bias} = +(1.111)2^{6-3} = 15$

b) Infinito:

0	1	1	1	0	0	0

$= +(1.000)2^{7-3} = 16$

c) Uno:

0	0	1	1	0	0	0

$= +(1.000)2^{3-3} = 1$

d) NaN

0	1	1	1	0	0	1

$= +(1.001)2^{7-3} = 18$

e) Número más pequeño normalizado Ei=0

0	0	0	1	0	0	0

$= +(1.000)2^{1-3} = 0.25$

f) Sistema: β=base, t=precisión, L= mínimo valor de Ee U= máximo valor de Ee
F(β,t,L,U)= (2,3,-2,3)

g) Cardinalidad= $2(\beta-1)\,\beta^{t-1}(U-L+1)+1$ =49 incluido el cero.

Ejemplo 15

Dado el siguiente número expresado en formato IEEE 754 de simple precisión:
0 10011011 00000000000000000000000
A que decimal representa?

Solución

$(1.0)*2^{155-127} = 2$

Ejemplo 16
Aritmética del computador en MATLAB

Considere la variable numérica de MATLAB

```
>> t = 0.1
```

La función `sym` tiene cuatro opciones para retornar a una representación simbólica del valor numérico almacenado en t .
La opción '`f`' :retorna una representación simbólica de punto flotante '

```
>> sym(t, 'f' )

 '1.999999999999a'*2^(-4)
```

La opción '`e`' retorna la forma racional de `t` más la diferencia entre la expresión racional teórica para `t` y su valor en punto flotante en términos de eps .
(la exactitud relativa del punto flotante):

```
>>sym(t, 'e')

ans =
1/10+eps/40
```

La opción ' r ' retorna la forma racional

```
>>sym(t, 'r' )
       1/10
```

Ésta es la forma fijada por defecto para sym..
La opción 'd' vuelve la extensión decimal de t hasta el número de dígitos significativos especificados por digits

```
>>sym(t, 'd' )
.1000000000000000055511151231257 8
```

El valor prefijado de dígitos es 32 (por lo tanto, sym(t, 'd ') retorna un número con 32 dígitos significativos), pero si usted prefiere una representación más corta, use el comando digits como sigue:

```
>>digits(7)
sym(t, 'd ')

ans =
1000000
```

El MATLAB usa un almacenamiento de doble precisión usando un total de 64 bits, para ver como se almacena un número en formato hexadecimal:
» format hex
» t
t =
3fb999999999999a
Los 13 últimos dígitos corresponden a la mantisa y los tres primeros corresponden al signo y exponente:

Donde $1023+e=1023+(-4)=1019=3x16^2+15*16+11=3fb_{(16)}$

Precisión aritmética de las variables (vpa) *variable presicion arithmetic*
Sintaxis:

```
R = vpa(A)
R = vpa(A,d)
```

Para calcular cada elemento de A con *d* dígitos decimales de exactitud, cada elemento del resultado es una expresión simbólica.
A=hilb(3);

```
R=vpa(A,4)
>> R =

   [    1., .5000, .3333]
   [ .5000, .3333, .2500]
   [ .3333, .2500, .2000]
```

Convirtiendo de simbólico a punto flotante
Para convertir una variable simbólica a numérica en MATLAB, use la función: **double**.

Ejercicios Propuestos

1. Desarrolle una función llamada "**nt"**, que retorne el número de términos necesarios para aproximar el número π hasta n cifras decimales exactos, usando la siguiente serie:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9}.....\right)$$

Solución

```
%n=cifras significativas
function y=nt(n)
y=1;
error=1;
t=4;
s=t;
while ((error)>(10^-n))
    ts=4*((-1)^y)/(2*y+1);
    ……..
end
```

2. Donde debe de estar x* para que aproxime a 1000 con 4 cifras significativas

.....
.....

3. Calcular sen(x+ ε)-sen(x) para valores pequeños de ε

.....
.....

4. Crea una **función** que represente el tiro parabólico en tres dimensiones, sabiendo que las coordenadas vienen dadas por las ecuaciones: x=Vo cos(θ) cos(φ) t ; y= Vo cos(θ) sin(φ) t ; z= Vo sin(θ) t-(0.5 g t^2); siendo θ el ángulo inicial que forma con la vertical y φ el ángulo inicial que forma con el eje X.

5. Complete la siguiente tabla de conversiones

Numero (Base 10)	Numero (Base2)
123	
124.25	
11.23	
-29.625	
0.1	

6. Representar en punto flotante con 32 bits, 1 de signo, 7 de exponente y 24 de mantisa, los números decimales 104.3125 y -13506.96875.

7. ¿Cuál es el valor decimal de: 1001111100011110?

8. Sea los siguientes comandos en MATLAB:

» format hex
» T
3fb0000000000000
El valor decimal de T es:

9. El producto **realmin*eps** en MATLAB representa a:
   a. Menor valor positivo Normalizado
   b. Menor valor positivo No Normalizado
   c. Mayor valor positivo Normalizado
   d. Mayor valor positivo No Normalizado

10. El **realmin** en MATLAB representa a:
    a. Menor valor positivo Normalizado
    b. Menor valor positivo No Normalizado
    c. Mayor valor positivo Normalizado
    d. Mayor valor positivo No Normalizado

11. Desarrolle una función llamado arcsen, que calcule el arcsen(x) y la diferencia con el valor exacto (asin(x)), a partir de la serie de Taylor, considerar n términos de dicha serie. Solo deberá calcular si x esta en el dominio apropiado sino tendrá que mandar un mensaje de error y grabar nan en las variables de salida.

$$arcsen(x) = \sum_{i=0}^{\infty} \frac{(2i)!}{4^i (i!)^2 (2i+1)} x^{2i+1} \qquad \text{para todo } |x| < 1$$

Escriba un programa que evalúe la serie:

$S = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} \cdots \frac{x^n}{n!}$ **. Probar para x=2 y n=10.**

	Curso	**METODOS NUMERICOS**	**Código : MB536**
	Tema	**Métodos Directos de Sistema de Ecuaciones Lineales (SEL)**	
	Practica	**03**	
	Profesores	**Garrido Juárez, Rosa** **Castro Salguero, Robert** **Pantoja Carhuavilca Hermes** **Obregón Ramos, Máximo** **Ruiz Lizama Edgar**	

1. **Objetivo :**
   Aplicar los métodos directos de factorización y eliminación en la solución de sistemas lineales.

2. **Fundamento Teórico**
   Sea el sistema de Ecuaciones Lineales:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\cdots$$
$$\cdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = bn$$

   O en forma matricial:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix}}_{b}$$

   Puede resolver sistemas lineales $Ax = b$ usando la inversa de la matriz A:

```
>> x=inv(A)*b
```

   Pero es numéricamente ineficiente comparada con otros procedimientos para calcular la inversa conocidos como factorizaciones que dan origen a los métodos directos.

   Métodos Directos

   En MATLAB se utiliza el comando \ para denotar la inversa por la izquierda, por lo cual para resolver un sistema de ecuaciones lineales (SEL) se ejecutaría el siguiente Comando:

```
>> x=A\b
```

   Métodos Directos: **Factorizaciones** A=L*U $\Leftarrow$ $a_{ij} = \sum_{k=1}^{n} l_{ik} u_{kj}$

   Matriz Triangular Inferior

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & & 0 \\ \vdots & & & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{11} \end{bmatrix} \qquad U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & & u_{2n} \\ \vdots & & & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix}$$

   Matriz Triangular Superior

   Clases de Factorizaciones:

- Doolitle $l_{ii}=1\ \forall i$ **(Eliminación Gaussiana)**
- Crout $u_{ii}=1\ \forall i$ **preferencia matrices tridiagonales**
- Cholesky $A=LL^T$ **A simétrica y definida positiva**

Aplicando en el sistema lineal:

$Ax=b \quad A=LU$

$$L\underbrace{Ux}_{z}=b$$

Formando dos sistemas triangulares fáciles de resolver:

$Lz=b$ (1)

$Ux=z$ (2)

La ecuación (1) se resuelve usando **sustitución directa** y la ecuación (2) usando la **sustitución inversa**.

Método Directo - Eliminación Gaussiana con o sin Pivoteo, consiste en trasformar el sistema aumentado en una matriz triangular superior y luego resolver el sistema triangular aplicando sustitución inversa.

Sistema:Lineal -	**Forma Matricial**	**Aumentado**
$a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1$ $a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=b_2$ $\cdots$ $\cdots$ $a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=bn$	Ax=b $\underbrace{\begin{bmatrix}a_{11}&a_{12}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2n}\\\vdots&&&\vdots\\a_{n1}&a_{n2}&\cdots&a_{nn}\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}x_1\\x_2\\\cdots\\x_n\end{bmatrix}}_{x}=\underbrace{\begin{bmatrix}b_1\\b_2\\\cdots\\b_n\end{bmatrix}}_{b}$	[A b] $\left[\begin{array}{cccc\|c}a_{11}&a_{12}&\cdots&a_{1n}&b_1\\a_{21}&a_{22}&\cdots&a_{2n}&b_2\\\vdots&&&\vdots&\cdots\\a_{n1}&a_{n2}&\cdots&a_{nn}&b_n\end{array}\right]$
Eliminación Gaussiana.-Triangulación: $\left[\begin{array}{cccc\|c}a_{11}&a_{12}&\cdots&a_{1n}&a_{1n+1}\\a_{21}&a_{22}&\cdots&a_{2n}&a_{2n+1}\\\vdots&&&\vdots&\cdots\\a_{n1}&a_{n2}&\cdots&a_{nn}&a_{nn+1}\end{array}\right]$ Operaciones x filas Pivote = $a_{ii}\neq 0$ $lik=\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$ for $k<i\leq n$ $a_{ij}^{(k+1)}=a_{ij}^{(k)}-lik*a_{kj}^{(k)}$ for $j<k<=n$ y $k<i\leq n$	Resultados de 1era Etapa (k=1) $\left[\begin{array}{cccc\|c}a_{11}&a_{12}&\cdots&a_{1n}&a_{1n+1}\\0&a_{22}^{(2)}&\cdots&a_{2n}^{(2)}&a_{2n+1}^{(2)}\\\vdots&&&\vdots&\cdots\\0&a_{n2}^{(2)}&\cdots&a_{nn}^{(2)}&a_{nn+1}^{(2)}\end{array}\right]$ pivot= a_{11} $li1=\frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$ for $k=1, i=2,..n$ $a_{ij}^{(2)}=a_{ij}^{(1)}-li1*a_{1j}^{(1)}$ for $i=2,..n \quad j=1,..n+1$	Resultados k=n-1 $\left[\begin{array}{cccc\|c}a_{11}&a_{12}&\cdots&a_{1n}&a_{1n+1}\\0&a_{22}^{(2)}&\cdots&a_{2n}^{(2)}&a_{2n+1}^{(2)}\\\vdots&&&\vdots&\cdots\\0&0&\cdots&a_{nn}^{(n-1)}&a_{nn+1}^{(n-1)}\end{array}\right]$ pivot= $a^{(k)}{}_{n-1n-1}$ $lik=\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$ for $k=n-1, \quad i=n$ $a_{nj}^{(k+1)}=a_{nj}^{(k)}-lnk*a_{kj}^{(k)}$ for $i=n \quad j=n,n+1$
$\underbrace{\begin{bmatrix}u_{11}&u_{12}&\cdots&u_{1n}\\0&u_{22}&\cdots&u_{2n}\\\vdots&&&\vdots\\0&0&\cdots&u_{nn}\end{bmatrix}}_{U}\underbrace{\begin{bmatrix}x_1\\x_2\\\cdots\\x_n\end{bmatrix}}_{x}=\underbrace{\begin{bmatrix}c_1\\c_2\\\cdots\\c_n\end{bmatrix}}_{c}$	**Sustitución Inversa** $x_k=\frac{c_k-\sum_{j=k+1}^{n}u_{k,j}x_j}{u_{k,k}}$	Solución $x=\begin{bmatrix}x_1\\x_2\\\cdots\\x_n\end{bmatrix}$

3. Instrucciones básicas en MATLAB

Muchas matrices especiales son funciones predefinidas; entre ellas están:

zeros(m,n)	Crea la matriz nula de orden mxn
ones(m,n)	Crea la matriz de orden mxn con todos sus elementos 1
rand(m,n)	Crea una matriz aleatoria uniforme de orden mxn
randn(m,n)	Crea una matriz aleatoria normal de orden mxn
flipud(A)	Devuelve la matriz cuyas filas están colocadas en orden inverso (de arriba abajo) a las filas de A
fliplr(A)	Devuelve la matriz cuyas columnas están colocadas en orden inverso (de izquierda a derecha) a las de A
rot90(A)	Rota 90 grados la matriz A
reshape(A,m,n)	Devuelve la matriz de orden mxn obtenida a partir de la matriz A, tomando elementos consecutivos de A por columnas
tril(A,k)	Extrae la parte triangular inferior de A debajo k_ésima diagonal
triu(A,k)	Extrae la parte triangular superior de A sobre la k_ésima diagonal

Resolución de sistemas

solve('ecuación', 'x')	Resuelve la ecuación en la variable x
solve ('ex1,ex2,...,exn', 'x1,x2,...,xn')	Resuelve n ecuaciones simultáneas ec1,...,ecn en las variables $x_1,...,x_n$ (sistema de ecuaciones)
X=linsolve(A,B)	Resuelve A*X =B para una matriz cuadrada A, siendo B y X matrices
roots(V)	Da las raíces del polinomio cuyos coeficientes son las componentes del vector V.
X=A\B	Resuelve el sistema A*X=B
X=A/B	Resuelve el sistema X*A=B

Operaciones Lógicas con Matrices

any(*v*)	Devuelve 0 si todos los elementos del vector *v* son nulos, y devuelve 1 si alguno de los elementos de *v* es no nulo.
all(*v*)	Devuelve 1 si todos los elementos del vector *v* son no nulos, y devuelve 1 si alguno de los elementos de *v* es nulo.
find(*v*)	Devuelve los lugares (ó índices) que ocupan los elementos no nulos del vector *v*.

Factorizaciones

[L,U]=lu(A) [L,U,P]=lu(A) R=chol(A) [Q,R]=qr(A)	Descompone la matriz A en el producto A =L*U (descomposición LU de A), siendo U una matriz triangular superior y L una matriz pseudos triangular inferior (triangularizable mediante permutación). Devuelve una matriz triangular inferior L, una matriz triangular superior U, y una matriz de permutación P tales que P*A =L*U. Devuelve la matriz triangular superior R tal que R´*R=A (Descomposición de Cholesky de A), en caso de que A sea definida positiva. Si A no es definida positiva devuelve un error. Devuelve la matriz triangular superior R de la misma dimensión que A, y la matriz ortogonal Q tales que A=Q*R(descomposición QR de A). Esta descomposición puede aplicarse a matrices no cuadradas.

4. Parte práctica

Dada la siguiente matriz: $A=\begin{bmatrix} 11 & -13 & 4 \\ -22 & 4 & -8 \\ 6 & 8 & 10 \end{bmatrix}$

Instrucción	**Solución**
Hallar $\|A\|_\infty$ manualmente	34
Hallar $\|A\|_\infty$ por definición usando comandos de MATLAB	max(sum(abs(A')))
Hallar $\|A\|_2$ con un comando de MATLAB	norm(A,2)
Hallar radio espectral $\rho(A)$ usando comandos de MATLAB	max(abs(eig(A)))
Hallar $\|A\|_2$ por un comando de MATLAB	norm(A,2)
Hallar $\|A\|_2$ por definición usando comandos de MATLAB	max(abs(eig(A*A')))^0.5

Factorización LU con pivoteo en MATLAB

_

Comando: [L,U]=lu(A)	

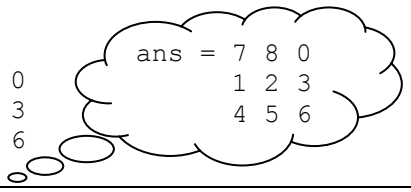
L es una matriz **triangular inferior**
U es una matriz **triangular superior**
 	
L*U = A
	`>> A=[1 2 3;4 5 6;7 8 0];`
	`>> [L,U]=lu(A)`
	`L = 0.1429 1.0000 0`
	`0.5714 0.5000 1.0000`
	`1.0000 0 0`
	`U = 7.0000 8.0000 0`
	`0 0.8571 3.0000`
	`0 0 4.5000`
	`>> L*U`
	`ans = 1 2 3`
	`4 5 6`
	`7 8 0`

Factorización LU con pivoteo en MATLAB

Comando: [L,U,P]=lu(A)
L es una matriz triangular inferior
U es una matriz triangular superior
P es una matriz de permutación

L*U = P*A.

```
>> A=[1 2 3;4 5 6;7 8 0];
>> [L,U,P]=lu(A)

L = 1.0000 0 0
    0.1429 1.0000 0
    0.5714 0.5000 1.0000

U = 7.0000 8.0000 0
         0 0.8571 3.0000
         0      0 4.5000
P = 0 0 1
    1 0 0
    0 1 0
>> L*U
ans = 7 8 0
      1 2 3
      4 5 6
>> P*A
```



Ejemplo de la factorización de Doolitle

$$\begin{bmatrix} 25 & 5 & 4 \\ 10 & 8 & 16 \\ 8 & 12 & 22 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Cuál de las opciones nos daría la correcta matriz L de la factorización de Doolitle?

(A) $\begin{bmatrix} 25 & 5 & 4 \\ 0 & 6 & 14.400 \\ 0 & 0 & -4.2400 \end{bmatrix}$

(B) $\begin{bmatrix} 1 & 0 & 0 \\ 0.40000 & 1 & 0 \\ 0.32000 & 1.5000 & 1 \end{bmatrix}$

(C) $\begin{bmatrix} 1 & 0 & 0 \\ 0.40000 & 1 & 0 \\ 0.32000 & 1.7333 & 1 \end{bmatrix}$

(D) $\begin{bmatrix} 1 & 0 & 0 \\ 10 & 1 & 0 \\ 8 & 12 & 0 \end{bmatrix}$

Respuesta: (C)

Ejemplo de la Factorización de Crout

Para la misma matriz anterior cual es la matriz U de la factorización de Crout

(A) $\begin{bmatrix} 25 & 5 & 4 \\ 0 & 6 & 14.400 \\ 0 & 0 & -4.2400 \end{bmatrix}$

(B) $\begin{bmatrix} 1 & 0.2000 & 0 \\ 0 & 1 & 2.6667 \\ 0 & 0 & 1 \end{bmatrix}$

(C) $\begin{bmatrix} 1 & 0.4000 & 0.3200 \\ 0 & 1 & 1.7333 \\ 0 & 0 & 1 \end{bmatrix}$

(D) $\begin{bmatrix} 1 & 10 & 8 \\ 0 & 1 & 12 \\ 0 & 0 & 0 \end{bmatrix}$

Respuesta: (B)

Algoritmo de Cholesky (Matrices A definidas positivas y Simétricas)

For k=1,2…..n do

For i=1,2,…..,k-1 do

$$a_{ki}=h_{ki}=\frac{1}{h_{ii}}\left(a_{ki}-\sum_{j=1}^{i-1}h_{ij}h_{kj}\right)$$

$$a_{kk}=h_{kk}=\sqrt{a_{kk}-\sum_{j=1}^{k-1}h_{kj}^2}$$

Nota en este algoritmo h_{ki} y h_{kk} se refiere a l_{ki} y l_{kk} respectivamente.

Ejemplo de la Factorización de cholesky

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 14 \\ 3 & 14 & 41 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ \ell_{21} & l_{22} & 0 \\ \ell_{31} & \ell_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ 0 & l_{22} & l_{23} \\ 0 & 0 & l_{33} \end{bmatrix}$$

Cuál de las opciones nos daría la correcta matriz L de la factorización de Cholesky?

(A) $\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 1.7333 & 1 \end{bmatrix}$

(B) $\begin{bmatrix} 3 & 1 & 1 \\ 0 & 1 & 8 \\ 0 & 0 & \sqrt{41} \end{bmatrix}$

(C) $\begin{bmatrix} 1 & 0 & 0 \\ 2 & 8 & 0 \\ 3 & 4 & \sqrt{41} \end{bmatrix}$

(D) $\begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & 4 & 4 \end{bmatrix}$

Respuesta (D)

Ejemplo de Eliminación Gaussiana:

Sea el sistema lineal

$$x_1 + 2x_2 + 3x_3 = 27$$
$$5x_1 - x_2 + 4x_3 = 47$$
$$-3x_2 + 2x_3 = 11$$

O en forma matricial:

Paso1: El "elemento Pivot" es a_{11}=1

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & -1 & 4 \\ 0 & -3 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 27 \\ 47 \\ 11 \end{bmatrix}$$

Paso 2: Eliminando el elemento a_{21} = 5 y modificando los elementos remanentes de la fila 2. Para hacer esto, multiplicamos los elementos de la fila 1 por $\boxed{a_{21}/a_{11} = 5}$ y restamos de la fila 2. Esto nos da:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -11 & -11 \\ 0 & -3 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 27 \\ -88 \\ 11 \end{bmatrix}$$

Paso 3: Eliminamos los elementos remanentes debajo del pivot y modificamos los elementos de las filas remanentes. En este caso a_{31} es cero. En este punto, tenemos:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -11 & -11 \\ 0 & -3 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 27 \\ -88 \\ 11 \end{bmatrix}$$

Paso 4 Continúe para dar la forma de matriz triangular por seleccionando los elementos pivotes, eliminando los elementos debajo del pivote, y modificando los elementos remanentes. Esto da:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -11 & -11 \\ 0 & 0 & 5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 27 \\ -88 \\ 35 \end{bmatrix}, \text{ o } \left[\begin{array}{ccc|c} 1 & 2 & 3 & 27 \\ 0 & -11 & -11 & -88 \\ 0 & 0 & 5 & 35 \end{array}\right]$$

Donde la última matriz es una matriz triangular superior (o en forma aumentada).

La solución de esta matriz triangular superior es usando sustitución inversa.

Step 5. **La sustitución hacia atrás o inversa:**

$X_3 = 35 / 5 = 7$
$X_2 = (-88 + 11x7) / (-11) = 1$
$X_1 = (27 - 2x1 - 3x7) / 1 = 4$

El vector solución es:

$$x = \begin{bmatrix} 4 \\ 1 \\ 7 \end{bmatrix}$$

5. Ejercicios Propuestos

Resuelva, con la ayuda de MATLAB, los siguientes problemas:

1. Escriba una función de MATLAB llamada **menores**,
   function [y]=menores(A,k)
   Variables de entrada: A, *k*
   Variable de salida : y que devuelva la sub-matriz cuadrada $k \times k$ de la matriz *A* correspondiente al menor principal de ese orden.

```
function [y]=menores(A,k)

y=……;
```

   Recuerde que un criterio suficiente para que una matriz *A* sea **definida positiva** es que **todos sus menores principales** sean **estrictamente positivos**. Modifique la función **menores** del ejercicio anterior para que devuelva la lista de todos los menores principales de la matriz argumento. Use la función **menores** para verificar si las matrices:

$$\begin{matrix} 21 & -13 & 2 \\ 13 & 133 & 14 \\ 2 & 14 & 5 \end{matrix} \quad \text{y} \quad \begin{matrix} 21 & -13 & -200 \\ -13 & 133 & 14 \\ -200 & 14 & 5 \end{matrix}$$

2. Crear la subrutina llamada *sustidir.m* que resuelva un sistema triangular inferior utilizando el algoritmo siguiente.

ENTRADA : Matriz triangular inferior L y vector b, tales que Lx=b
SALIDA : Vector x
Paso 1 : Verificar que el sistema es triangular inferior
Paso 2 : Obtener el orden del sistema, n
Paso 3 : Para k desde 1 hasta n repetir pasos 4-5
Paso 4: Verificar que el elemento de la diagonal no es nulo
Paso 5: Hacer $x_k = \dfrac{b_k - \sum_{j=1}^{k-1} L_{k,j} x_j}{L_{k,k}}$

```
function [x ]= susdir(L ,b)
if any(any(tril(L)-L)), error('no es mat. triangular inferior')
else
[n,m]=size(L); x=zeros(n,1);
for k=1:n
   j= 1:k-1
x(k)=(b(k)-L(k,j)*x(j))/L(k,k);
end
```

Para probar: $L=\begin{bmatrix}1&0&0\\2&3&0\\1&2&3\end{bmatrix}$ y $b=\begin{bmatrix}1\\2\\4\end{bmatrix}$, y la matriz $L=\begin{bmatrix}1&1&1\\2&3&1\\1&2&3\end{bmatrix}$ y $b=\begin{bmatrix}1\\1\\1\end{bmatrix}$

```
>> x= susdir(  L ,b  )
```

3. Crear la subrutina llamada *sustinv.m* que resuelve un sistema triangular superior utilizando el algoritmo discutido en clase.

   ENTRADA : Matriz triangular inferior U y vector c, tales que Ux=c
   SALIDA : Vector x
   Paso 1 : Verificar que el sistema es triangular superior
   Paso 2 : Obtener el orden del sistema, n
   Paso 3 : Para k desde n hasta 1 repetir pasos 4-5 (k=n:-1:1)
   Paso 4: Verificar que el elemento de la diagonal no sea nulo
   Paso 5: Hacer $x_k=\dfrac{b_k-\sum_{j=k+1}^{n}u_{k,j}x_j}{u_{k,k}}$

4. Crear la Subrutina llamada Gauss sin intercambio de filas utilizando el algoritmo discutido en clase.

```
function  [x]= gauss (A,b)
% Elaborado por Rosa Garrido
% Primera versión sin pivoteo
% parámetros de entrada : A y b (vector columna)
% Los comandos que ejecuta la función "gauss.m" son los siguientes:

n = length(b);              % n guarda el número de elementos del vector b.
for k = 1:(n-1)             % El ciclo for comienza en k=1 hasta k=n-1
    i = k+1:n               % valor del vector i dedes k+1 hasta n
m= A(i,k)/A(k,k)            % m guarda el valor del elemento modificador
A(i,:) = A(i,:) - m*A(k,:); % modificando el bloque debajo de la diagonal
b(i)= b(i) - m*b(k);        % modificando el vector b
end                         % Finaliza el ciclo del for de variable k
x=zeros(n,1);               % inicializando el vector x
for k = n:-1:1              % Comienza el ciclo de la sustitución en inversa,
x(k) = (b(k) - A(k,k+1:n)*x(k+1:n))/A(k,k);
end                         %Finaliza el ciclo del for de variable k
```

5. Modifica la rutina de gauss.m incluyendo en ella la subrutina de sustitución inversa.
   Incorporar a la subrutina anterior una estrategia de pivoteo parcial.

6. Crear la Subrutina llamada Crout que permita resolver el sistema lineal **Ax=b** debe comprobar primero si A es una matriz tridiagonal., en caso contrario enviar mensaje de fracaso.

Inicio $l_{11} = a_{11}$ $u_{12} = a_{12}/l_{11}$ Para i = 2 hasta n - 1 hacer $l_{i,i-1} = a_{i,i-1}$ $l_{ii} = a_{ii} - l_{i,i-1}u_{i-1,i}$ $u_{i,i+1} = a_{i,i+1}/l_{ii}$ Fin_Para $l_{n,n-1} = a_{n,n-1}$ $l_{nn} = a_{nn} - l_{n,n-1}u_{n-1,n}$ s_{i+1} Fin_Para Fin	function [L,U] = crout(A) % Probar si A es una matriz

7. Crear la Subrutina llamada Cholesky que permita resolver el sistema lineal **Ax=b**, comprobando primero si A es una matriz simétrica, en caso contrario enviar mensaje de fracaso.
8. Obtener la factorización de Cholesky de la matriz

$$A = \begin{bmatrix} 4 & 2 & 2 & 4 \\ 2 & 5 & 7 & 0 \\ 2 & 7 & 19 & 11 \\ 4 & 0 & 11 & 25 \end{bmatrix}$$

   a. Manualmente
   b. Utilizando la función **chol** de MATLAB

9. Crear la Subrutina llamada Doolitle que permita resolver el sistema lineal **Ax=b**, usando la factorización LU

$$A = \begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} \xrightarrow[m_{31}=4]{m_{21}=-2} \begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 6 & -3 \end{bmatrix} \xrightarrow{m_{32}=3} \begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix}$$

Notemos que:

$$\text{L*U} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{bmatrix} = \begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} = A$$

10. Resolver el sistema de ecuaciones lineales $A \cdot \vec{x} = \vec{b}$, siguiente:

$$\begin{bmatrix} 5 & -3 & 2 \\ -3 & 8 & 4 \\ 2 & 4 & -9 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 9 \end{bmatrix}$$

a. Usando la function inv del MATLAB.
b. Compare sus resultados usando A^{-1}, compare el resultado de usar la función inv. ¿Qué es lo que Ud. concluye?
c. ¿Qué valor tiene el producto: $A \cdot A^{-1}$?
d. ¿Cuál es el resultado esperado a partir de $A \cdot A^{-1} - A^{-1} \cdot A$?

11. Realizar la factorización LU de la matriz

$$A = \begin{bmatrix} 8 & 2 & 9 \\ 4 & 9 & 4 \\ 6 & 7 & 9 \end{bmatrix}$$

a. Sin Pivoteo Parcial
b. Con Pivoteo Parcial

12. Si se pretende resolver el sistema *Ax=b* de forma óptima, con A simétrica y definida positiva. Cuál de los siguientes procesos es más óptimo.

a. Primero se calcula la descomposición LU y luego se resuelven los sistemas triangulares.
b. Primero se calcula la descomposición LU con pivoteo parcial y luego se resuelve los sistemas triangulares.
c. Se calcula la descomposición de Cholesky y luego se resuelven los sistemas triangulares.
d. Se calcula inicialmente la descomposición de Cholesky y a continuación se resuelve el sistema triangular inferior, cuya solución coincide con la del sistema Ax=b.

13. Sea la matriz *A*:

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 3 & 1 & 0 \\ 4 & 2 & k \end{bmatrix}$$

A través de factorización *LU* determine a matriz inversa de A.
Determine el número de condicionamiento de la matriz A.

a. $cond(A) = \|A\|\|A^{-1}\|$ para $\|A\| = \max_i \sum_j l_{ij}$ y $0 < k \leq 1$.

	Curso	**Métodos Numéricos**	**Código : MB536**
	Tema	**Métodos Iterativos para Resolver Sistemas de Ecuaciones Lineales**	
	Practica	**04**	
	Profesores	**Garrido Juárez, Rosa** **Castro Salguero, Robert** **Pantoja Carhuavilca, Hermes** **Obregón Ramos, Máximo** **Ruiz Lizama, Edgar**	

1. Objetivos

Estudiar los métodos iterativos para resolver sistemas de ecuaciones lineales a partir de un vector inicial, generando una sucesión de vectores que deben converger bajo ciertas condiciones a la solución en un número de iteraciones finitas. También se estudian métodos iterativos para el cálculo de valores y vectores característicos.

2. Fundamento teórico

2. 1 Métodos Iterativos para la solución de Sistemas de Ecuaciones Lineales

Los métodos iterativos para resolver sistemas lineales de la forma: $A\,x = b$, pueden expresarse como $\mathbf{x}^{(k+1)} = \mathbf{T}\mathbf{x}^{(k)} + \mathbf{c}$. La matriz **T** y vector **c** varían de acuerdo al método. Además **A = D – L – U**.

Donde:

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ \vdots & a_{22} & & \vdots \\ & & \ddots & \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}, L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix}, U = \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Método	*T*	*c*
Jacobi	$T_j = D^{-1}(L+U)$	$c_j = D^{-1}b$
Gauss-Seidel	$T_g = (D-L)^{-1}U$	$c_g = (D-L)^{-1}b$

2.2 Convergencia de los métodos iterativos

Teorema 1.- Para cualquier $x^{(0)} \in \Re^n$, la sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ definida por

$x^{(k)} = Tx^{(k-1)} + c$, para $k \geq 1$ y $c \neq 0$,

converge a la solución única de $x = Tx + c$ si y solo si $\rho(T) < 1$.

Teorema 2.- Si ***A*** es estrictamente dominante diagonalmente, entonces, para cualquier elección de $x^{(0)}$ ambos métodos, el de Jacobi y el de Gauss-Seidel, dan lugar a sucesiones $\{x^{(k)}\}_{k=0}^{\infty}$ que convergen a la solución de ***Ax=b***.

2.3 Métodos Iterativos para el Cálculo de Valores y Vectores Propios

Método de la Potencia

Definición

Sea λ_1 un autovalor de **A** que en valor absoluto es mayor que cualquier otro autovalor, entonces se dice que es un **autovalor dominante** *y su* autovector correspondiente se llama **autovector dominante.**

Algoritmo de la Potencia

proc potencia(input: A, z_0, ϵ, $k_{\max}$; *output:* λ_k, z_k)
    $z_0 \leftarrow z_0 / || z_0 ||$
    $\lambda_0 \leftarrow z_0^T A z_0$
    for $k = 1, 2, \ldots, k_{\max}$ *do*
        $q \leftarrow A z_{k-1}$
        $z_k \leftarrow q / || q ||$
        $\lambda_k \leftarrow z_k^T A z_k$
        if $| \lambda_k - \lambda_{k-1} | < \epsilon$ *then*
            break
        endif
    endfor
endproc

3. Instrucciones básicas en Matlab

Instrucción	Descripción
diag(A)	Diagonal de la Matriz A
triu(A)	Parte triangular Superior de la Matriz A
tril(A)	Parte triangular Inferior de la Matriz A
eig(A)	Valores propios de la Matriz A
inv(A)	Inversa de la Matriz
norm(x,2)	Norma Euclidiana de x, $\|x\|_2$
norm(x,inf)	Norma infinita de x, $\|x\|_\infty$

4. Parte Practica

1. Implementar el método de Jacobi y resolver el siguiente sistema:

$$\begin{aligned} 4x_1 + 0.24x_2 - 0.08x_3 &= 8 \\ 0.09x_1 + 3x_2 - 0.15x_3 &= 9 \\ 0.04x_1 - 0.08x_2 + 4x_3 &= 20 \end{aligned}$$

```
function [z,x,numite]=jacobi(A,b,TOL,MAXITE)
D = diag(diag(A));
L = D - tril(A); U = D - triu(A); Tj = inv(D)*(L+U);
x = zeros(size(b)); % Vector Inicial
Cj = inv(D)*b;  z = [];
for i = 1:MAXITE
        xn = Tj*x+Cj;
        err = norm(xn-x,2);
        z = [z;xn' err];
        x = xn;
        if err<TOL
            break
        end
end
 numite = i;
```

» A = [4 0.24 -0.08; 0.09 3 -0.15; 0.04 -0.08 4]
» B = [8; 9; 20]
» [z,x,numite]=jacobi(A,B,1e-6,100)

2. Use la función "eig" para encontrar los valores y vectores propios de la matriz

$$A = \begin{bmatrix} 3 & 2 & 2 \\ 1 & 4 & 1 \\ -2 & -4 & -1 \end{bmatrix}$$

Solución:

»A = [3 2 2;1 4 1;-2 -4 -1]
»[Q,D] = eig(A)
Q =

0.8944 0.7071 -0.0000
0.4472 0 0.7071

0.0000 -0.7071 -0.7071
D =

2.0000 0 0
0 1.0000 0
0 0 3.0000

Así los valores propios son λ = 2, 1, 3 y las columnas de Q corresponde a los vectores propios.

3. Implementar el método de la potencia para aproximar el valor propio dominante de la matriz $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 3 \\ 3 & 3 & 3 \end{bmatrix}$. Partiendo del vector $x_0 = (1 \quad 1 \quad 0)^T$

```
% potencia.m
% Metodo de la potencia para calcular el valor propio dominante y
% su vector correspondiente
function [z,l,x]=potencia(A,x0,MAXITE,TOL)
x=x0; z=[];
for i=1:MAXITE
   y=A*x;
   [m,p]=max(abs(y));
   l=y(p);
   err=norm(y/l-x,2);
   x=y/l;
   z=[z;x' l];
   if err<TOL
      break
   end
end
```

Utilizando MATLAB tenemos:

```
» A = [1 2 3 ; 2 2 3 ; 3  3  3]
» x0 = [1;  1;  0]
» [z,l,x]=potencia(A,x0,50,1e-5)
z =
   0.5000    0.6667    1.0000    6.0000
   0.7436    0.8205    1.0000    6.5000
   0.7000    0.7967    1.0000    7.6923
   0.7067    0.8002    1.0000    7.4900
   0.7057    0.7996    1.0000    7.5207
   0.7058    0.7997    1.0000    7.5159
   0.7058    0.7997    1.0000    7.5166
   0.7058    0.7997    1.0000    7.5165
l =
7.5165
x =
0.7058
0.7997
1.0000
```

Es decir, podemos concluir que una aproximación al valor propio de mayor valor absoluto con una tolerancia de 1e-5 es 7.5165 y su vector propio asociado es [0.7058 0.7997 1.0000].

4. (Tomado de Eduardo Raffo Lecca "*Métodos Numéricos para Ciencias e Ingeniería con* MATLAB", 2008)

   El método de **Gauss-Seidel** es también conocido como método de desplazamientos sucesivos.

   Resolver el sistema:

$$2x_1 + 10x_2 - x_3 = 11$$
$$20x_1 - x_2 + x_3 = 20$$
$$x_1 + x_2 - 20x_3 = 18$$

   Se observa, que aparentemente no son estrictamente dominantes en la diagonal, pero basta con realizar un intercambiando el orden de las ecuaciones, para conseguir la dominancia:

$$20x_1 - x_2 - x_3 = 20$$
$$2x_1 + 10x_2 - x_3 = 11$$
$$x_1 + x_2 - 20x_3 = -18$$

   La solución con *Gauss-Seidel*, con un error del 0.0005; se muestra en la figura 4.1.

```
>> GaussSeidel(a,b)
    0   0.00000   0.00000   0.00000
    1   1.00000   0.90000   0.99500
    2   1.09475   0.98055   1.00377
    3   1.09922   0.98053   1.00399
    4   1.09923   0.98055   1.00399
ans =

    1.0992
    0.9806
    1.0040
```

Figura 4.1: Ejecución de gaussSeidel.m

   El código Matlab para Gauss-Seidel es el siguiente:

```
function x=GaussSeidel(A,b)
%% Datos
% A = es la matriz
% b = es el vector de la mano derecha
% n = el orden de la matriz
% Resultados
% x = vector solucion
[n n] = size(A);
x = zeros(n,1);
y = zeros(n,1);
error = 0.0005;
flag = 1;
NTOL = 50;
k = 0;
fprintf('%5d',k);
for m = 1:n
    fprintf('%10.5f',x(m));
end
```

```
%  prueba de diagonalizacion
i = 1;
while (i <= n) & (flag == 1)
   suma = 0;
   for j = 1:n
       if i ~= j
          suma = suma + abs(A(i,j));
       end
   end
   if abs(A(i,i)) <= suma
      fprintf('\nError de diagonalizacion');
      flag = 0;
   end
   i=i+1;
end %     fin de la prueba

if flag == 0
    break
end

while 1
   flag = 1;
   for i = 1:n
          suma = 0;
          for j = 1:n
             if i ~= j
                suma = suma+A(i,j)*x(j)/A(i,i);
            end
          end
          y(i) = b(i)/A(i,i)-suma;
          if abs(y(i) - x(i)) > error
             flag = 0;
          end
          x(i) = y(i);
    end
    k=k+1;
     fprintf('\n%5d',k);
    for i=1:n
         fprintf('%10.5f',x(i));
    end
    if (NTOL == k) | (flag == 1)
         break
    end
end
```

5. Para resolver un cierto sistema 3×3 se obtuvo por Gauss-Seidel la matriz de iteración:

$$T_{G-S} = \begin{bmatrix} 0 & 1/2 & -1/3 \\ 0 & -2/3 & 1/6 \\ 0 & 0 & -1/9 \end{bmatrix}$$ y como vector de términos independiente: $c_{G-S} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

a) Tomando $x^{(0)} = [1,0,0]^t$, calcular $x^{(1)}$ y $x^{(2)}$. Dejar indicadas las operaciones matriciales.

b) ¿Las iteraciones de Gauss-Seidel convergen? Justificar.

Solución

a)

$$x^{(1)} = T_{G-S}x^{(0)} = \begin{bmatrix} 0 & 1/2 & -1/3 \\ 0 & -2/3 & 1/6 \\ 0 & 0 & -1/9 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x^{(2)} = T_{G-S}x^{(1)} = \begin{bmatrix} 0 & 1/2 & -1/3 \\ 0 & -2/3 & 1/6 \\ 0 & 0 & -1/9 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.1667 \\ 0.5 \\ 0.889 \end{bmatrix}$$

b)

$$\left|T_{G-S} - \lambda I\right| = \det\left(\begin{bmatrix} -\lambda & 1/2 & -1/3 \\ 0 & -2/3-\lambda & 1/6 \\ 0 & 0 & -1/9-\lambda \end{bmatrix}\right) = \lambda^3 - 7/9\lambda^2 + 2/27\lambda = 0$$

$\lambda = 0 \quad \lambda = -2/3 \quad \lambda = -1/9 \qquad \rho(Tgs) = 2/3 < 1 \quad \therefore$ Converge

6. Sea el sistema:

$$\begin{bmatrix} a & 1 \\ 1 & a+2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

a. Para qué valores de a, el método de Jacobi es convergente?
b. Realice 05 iteraciones de Jacobi, con $a = 1$, a partir de $x^{(0)}=[0,0]^t$
c. Cual es el error cometido? Comente sus resultados.

Solución

a)

$$Tj = \begin{bmatrix} 0 & -1/a \\ -1/(a+2) & 0 \end{bmatrix}$$

Calculo del radio espectral

$$\left|T_j - \lambda I\right| = \det\left(\begin{bmatrix} -\lambda & -1/a \\ -1/(a+2) & -\lambda \end{bmatrix}\right) = \lambda^2 - 1/(a^2+2a) = 0$$

$$\lambda = \sqrt{\frac{1}{a^2+2a}} < 1$$

$$a^2 + 2a - 1 > 0$$

$$\left(a-(-1-\sqrt{2})\right)\left(a-(-1+\sqrt{2})\right) > 0$$

a>-1+ √2= 0.4142

a=1

Sist. Lineal $\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ ➔ Algoritmo de Jacobi Iteraciones $x^{(0)}=[0\ \ 0]^t$

$$x^{(k+1)} = \begin{bmatrix} 0 & -1 \\ -1/3 & 0 \end{bmatrix} x^{(k)} + \begin{bmatrix} 1 \\ 1/3 \end{bmatrix}$$

$$x^{(1)} = \begin{bmatrix} 0 & -1 \\ -1/3 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/3 \end{bmatrix};\ x^{(2)} = \begin{bmatrix} 0 & -1 \\ -1/3 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1/3 \end{bmatrix} + \begin{bmatrix} 1 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 2/3 \\ 0 \end{bmatrix}$$

$$x^{(3)} = \begin{bmatrix} 0 & -1 \\ -1/3 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1/3 \end{bmatrix} + \begin{bmatrix} 2/3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/9 \end{bmatrix};\ x^{(4)} = \begin{bmatrix} 0 & -1 \\ -1/3 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1/9 \end{bmatrix} + \begin{bmatrix} 2/3 \\ 0 \end{bmatrix} = \begin{bmatrix} 8/9 \\ 0 \end{bmatrix}$$

$$x^{(5)} = \begin{bmatrix} 0 & -1 \\ -1/3 & 0 \end{bmatrix}\begin{bmatrix} 8/9 \\ 0 \end{bmatrix} + \begin{bmatrix} 2/3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/27 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.037 \end{bmatrix}$$

b) L = Radio espectral =máximo valor propio con a=1

$$\lambda = \sqrt{\frac{1}{a^2 + 2a}} < 1$$

L=0.5774

$$E \le \frac{L^5}{1-L}\left\| x^{(0)} - x^{(1)} \right\|_\infty = 0.1518$$

En la quinta iteración se ve que la solución converge al valor de $[1\ \ 0]^t$

5. Ejercicios propuestos

1. Crear una rutina en MATLAB para determinar si una matriz tiene diagonal estrictamente dominante.

function flag = dominante (A)

% flag : 1 si tiene diagonal estrictamente dominante y 0 en caso contrario

……..
…….....
…….
…….
……..

2. Crear una rutina en MATLAB para determinar si una matriz es simétrica, definida positiva y tridiagonal.

function flag = verifica (A)

% flag : 1 si A es simétrica, definida positiva y tridiagonal y 0 en caso contrario

……..
…….....
…….
…….

……..

3. Dado el sistema lineal:

$$\begin{bmatrix} a+2 & 1 \\ 1 & a \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

   a) Halle todos los valores de a que aseguren convergencia al aplicar el método de Jacobi.

   ……

   b) Con a=0.60, muestre la tercera iteración de Gauss-Seidel partiendo de $x_1^{(0)} = 0 \quad x_2^{(0)} = 0$:

   $x_1^{(3)} =$ …… $x_2^{(3)} =$ ……

4. Calcule todos los valores característicos de $M = \begin{pmatrix} 0 & -1 & -1 \\ -2 & 1 & -1 \\ -2 & 2 & 2 \end{pmatrix}$

   Realice 03 iteraciones del método de la potencia usando el método de la potencia inversa, a partir de $[1\ 1\ 1]^T$.

   ……..
   ……..
   ……
   ……
   ……..

5. Teniendo en cuenta que el método de Gauss – Seidel es convergente cuando la matriz A es simétrica y definida positiva encuentre para que valores de a es convergente la siguiente matriz:

$$A = \begin{pmatrix} \overbrace{\mathbf{1} & \mathbf{1} & \dots & \mathbf{1}}^{\mathbf{10\ veces}} & \mathbf{1} \\ \mathbf{1} & a & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & a-\mathbf{1} & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \dots & a-\mathbf{8} \end{pmatrix}$$

   a) $a > \mathbf{9}$ b) $-\mathbf{1} < a < \mathbf{1}$ c) $a > \mathbf{8}$ d) $-\mathbf{2} \le a \le \mathbf{2}$ e) $a > \mathbf{2}$

6. Sea: $A = \begin{bmatrix} -1 & 4 \\ 1 & -1 \end{bmatrix}$, ¿cuál de los siguientes es un vector propio de A?

   a) $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ b) $\begin{bmatrix} 1 \\ -2 \end{bmatrix}$ c) $\begin{bmatrix} 2 \\ -1 \end{bmatrix}$ d) $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ e) N.A.

7. Sea el sistema: $\begin{bmatrix} 10 & \frac{\alpha+5}{2} \\ \frac{\alpha+4}{2} & 20 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, si α es el último digito de su código entonces el radio espectral de Jacobí será: …….....

8. Considere la matriz $A = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 2 & 0 \\ 1 & a & a \end{bmatrix}$, *a* parámetro real

   a. Diga para que valores de *a*, la matriz admite el valor propio cero.
   b. Para ese valor de *a*, determine los restantes valores propios y los correspondientes vectores propios

9. Supóngase una matriz invertible de segundo orden y con elementos diagonales no nulos

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

   Hallar las matrices de iteración para los métodos de Jacobi y Gauss – Seidel e indicar el valor de verdad de las siguientes proposiciones:

   a. Ambos sistemas convergen cuando $\left|\frac{bc}{ad}\right| \geq 1$
   b. La convergencia del método de Gauss – Seidel es más rápida que Jacobi

10. Considerar el sistema lineal

$$\begin{pmatrix} 1 & 3 \\ 2 & -1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$$

    ¿Puede aplicarse Jacobi o Gauss-Seidel para resolver este sistema?. Encontrar un sistema equivalente al que si se le puede aplicar.

11. Dada la matriz

$$\begin{pmatrix} 1 & b & 0 \\ b & 2 & c \\ 0 & c & 1 \end{pmatrix}$$

    i. Probar que los métodos de Jacobi y de Gauss-Seidel convergen o divergen exactamente para los mismos valores de b y c.
    ii. En caso de converger. ¿Cuál lo hace más rápidamente?

	Curso	**Métodos Numéricos**	**Código : MB536**
	Tema	**Solución de Ecuaciones no Lineales de una y más variables**	
	Practica	**05**	
	Profesores	**Garrido Juárez, Rosa** **Castro Salguero, Robert** **Pantoja Carhuavilca, Hermes** **Obregón Ramos, Máximo** **Ruiz Lizama, Edgar**	

1. Objetivos :

Aplicar los métodos iterativos de intervalo y los métodos iterativos funcionales, en la solución de ecuaciones no lineales de una y más variables.

2. Fundamentos Teóricos

Métodos Iterativos de Intervalo

Método de Bisección

En la resolución de ecuaciones no lineales se utilizan, salvo soluciones analíticas simples, métodos iterativos que generan una sucesión de valores que tienden al valor de la raíz. Este método presenta la ventaja de acotar no sólo el valor de la función, sino también el intervalo a que pertenece la raíz. Para su aplicación es necesario que verifique las condiciones del Teorema de Bolzano, esto es, la función debe ser continua y cambiar de signo en sus extremos.

Algoritmo de Bisección

Dato: Leer ***a***, ***b*** tal que ***f*** es continua en [a,b] y f(a)*f(b)<0

```
para i=1 hasta MaxIte
    x=(a+b)/2;
    err=(b-a)/2;
    si f (a)*f (x)<0
        b=x;
    sino
        a=x;
    fin_si
    si err<TOL
        salir
    fin_si
fin_para
```

Métodos de Iteración Funcional

Continuando con los métodos de resolución de ecuaciones no lineales, analizaremos en esta práctica los métodos de iteración funcional, esto es, métodos que convierten la ecuación $f(x) = 0$ en $x = g(x)$. Estos métodos presentan la ventaja de poder calcular raíces de multiplicidad par (no existen intervalos en que la función cambie de signo), y el inconveniente de que no son capaces de acotar el intervalo a que pertenece la raíz (y por tanto es difícil evaluar su proximidad a la misma).

a) Método de Punto Fijo o Aproximaciones Sucesivas

Estos métodos se basan en el Teorema de Punto Fijo, y deben cumplir las condiciones del Teorema para garantizar su convergencia. Dichas condiciones se resumen como:

$g(x) \in C^1[a,b]$ $\forall x \in [a,b]: g(x) \in [a,b]$ $\forall x \in [a,b]: |g'(x)| < 1$

Algoritmo de aproximaciones sucesivas:

Sea $f(x) = 0$ *equivalente a* $x = g(x)$
Leer x_0
repetir para $n = 0, 1, \cdots$
$x_{n+1} = g(x_n)$
hasta $|x_{n+1} - x_n| < TOL$

Comenzaremos calculando por este método las raíces de $f(x)=x^4+2x^2-x-3$. El primer paso es su escritura en la forma $x=g(x)$. Utilizaremos cálculo simbólico para la representación y resolución de estas ecuaciones. Comenzaremos definiendo las variables y funciones simbólicas a emplear y representaremos las curvas.

```
» syms x                  % Definición de la variable simbólica x
» fx='x^4+2*x^2-x-3';     % Def. Función simbólica fx
» ezplot(fx,-2,2);grid on;    % Representación de fx en [-2,2]
```

Para encontrar las raíces de forma más exacta (simbólica), además de las funciones creadas en la práctica anterior, se puede utilizar 'solve' y 'fzero'. La primera necesita que se le de una ecuación, lo que se logra mediante:

```
» solve(strcat(fx,'=0'))
```

Para usar fzero, se le debe dar un punto próximo a la raíz o, mejor, un intervalo donde esta cambia de signo.

```
» x1=fzero(fx,[-1,0]),x2=fzero(fx,[0,2])
```

Despejando la primera x de la expresión, se tiene que
$g(x) = \sqrt[4]{3 + x - 2x^2}$ .Definiendo y representando la función $g(x)$ mediante

```
» gx=' (3+x-2*x^2)^0.25'; % Def. función simbólica gx
» ezplot(gx,-2,2);grid on; % Representación de gx en [-2,2]
```

Para que la sucesión de valores de punto fijo converja es necesario que se cumplan las condiciones enunciadas. La evaluación de la derivada de la función y su representación se logra usando:

```
» dgx=diff(gx); % Def. función simbólica dgx, derivada de gx
» ezplot(dgx,-2,2);grid on; % Representación de dgx en [-2,2]
```

b) Método de Newton-Raphson

A continuación utilizaremos el método de Newton, y veremos diferentes características del mismo. Este método se puede considerar una particularización del punto fijo, si bien existen condiciones suficientes de convergencia cuya demostración es más simple que en el caso de punto fijo.

$\boxed{f(x) \in C^2[a,b]}\ \forall x \in [a,b]: f'(x) \neq 0\ \boxed{\forall x \in [a,b]: f''(x)\ sig.\ cte}\ \boxed{\forall x \in [a,b]: \max|f(x)/f'(x)| \leq |b-a|}$

Algoritmo de Newton-Raphson:

Leer x_0

repetir para $n = 0, 1, \cdots$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

hasta $|x_{n+1} - x_n| < TOL$

SISTEMAS DE ECUACIONES NO LINEALES

Consideremos ahora el problema de resolver un sistema de ecuaciones no lineales de *n* ecuaciones con *n* variables. Sea $f_i : R^n \to R$, $1 \le i \le n$. funciones (no lineales) diferenciables. Un sistema no lineal $n \times n$ se puede escribir de la forma:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0, \\ f_2(x_1, x_2, \ldots, x_n) = 0, \\ \quad \vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0, \end{cases} \qquad (1)$$

Si definimos F: $R^n \to R^n$ por $F=(f_1,f_2,\ldots,f_n)^t$ entonces podemos escribir (1) en forma vectorial como:

$$F(\mathbf{x})=0, \quad \mathbf{x}=(x_1,x_2,\ldots,x_n) \qquad (2)$$

Método del punto fijo

Análogamente al caso unidimensional, consideremos nuevamente el sistema de ecuaciones $F(x)=0$, donde $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^t$ con $f_i : R^n \to R, \quad i = 1,2,\ldots,n$

El método iterativo del punto fijo se base en la posibilidad de escribir el sistema de ecuaciones $F(x)=0$ en otro equivalente de la forma

$$x = G(x)$$

Donde $G : \mathbf{R}^n \to \mathbf{R}^n$, osea:

$$\begin{cases} x_1 = g_1(x_1, x_2, \ldots, x_n) \\ x_2 = g_2(x_1, x_2, \ldots, x_n) \\ \quad \vdots \\ x_n = g_n(x_1, x_2, \ldots, x_n) \end{cases}$$

Donde $g_1, g_2, \ldots, g_n$ son los componentes de G

consiste entonces en generar una sucesión de puntos en $\mathbf{R}^n$ por medio de la relación de recurrencia

$$x^{(k)} = G(x^{(k-1)}), \qquad k = 1,2,\ldots,$$

a partir de un punto inicial $\boldsymbol{x}^{(0)}$. Se pretende que esta sucesión de puntos en $\mathbf{R}^n$ converja para un punto fijo *s* de la función ***G***, esto es, tal que *s* = ***G***(s) que será por tanto solución del sistema original, o sea, tal que ***F***(s)=**0.**

Método de Newton Raphson

El método de Newton para la solución de sistemas de ecuaciones es también una generalización del método ya estudiado para el caso unidimensional. Consideremos nuevamente el sistema de ecuaciones $F(x)=0$, donde

$$F(x) = \left(f_1(x), f_2(x), \ldots, f_n(x)\right)^t$$

con $f_i : R^n \to R, \quad i = 1,2,\ldots,n$

Definimos la matriz Jacobiana de la función F como:

$$J_F(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Sea $\boldsymbol{x_0}$ una aproximación inicial al sistema $F(\boldsymbol{x})=0$. Entonces usando el Teorema de Taylor para funciones de varias variables, podemos escribir que

$$F(x) \approx F(x_0) + J_F(x_0)(x - x_0)$$

Definimos ahora la siguiente aproximación $\boldsymbol{x_1}$ como la solución de

$$F(x_0) + J_F(x_0)(x - x_0) = 0$$

es decir

$$x_1 = x_0 - \left(J_F(x_0)\right)^{-1} F(x_0)$$

De esta forma continuamos así la versión para sistemas del *Método de Newton* dada por:

$$\begin{cases} x_{k+1} = x_k - \left(J_F(x_k)\right)^{-1} F(x_k), & k \geq 0 \\ x_0 \ \textit{dado} \end{cases}$$

La implementación del método de Newton para sistemas de ecuaciones no lineales

3. Instrucciones básicas en MATLAB

3.1Funciones para resolver ecuaciones no lineales de una y más variables.

En forma simbólica:

solve: Solución Simbólica de las ecuaciones algebraicas, su argumento puede ser una ecuación algebraica en cadena o un sistema de ecuaciones algebraicas.
ezplot : Graficador de funciones en cadena o en forma simbólica
p.e. ezplot('x^2-y^4'), puede especificarse el rango de x que se desea graficar.
eval: La función ***eval***('cadena de caracteres') hace que se evalúe como expresión de MATLAB el texto contenido entre las comillas como argumento de la función. Este texto puede ser un comando, una fórmula matemática o -en general- una expresión

válida de MATLAB. La función ***eval*** puede tener los valores de retorno necesarios para recoger los resultados de la expresión evaluada.

En forma numérica

fzero(fun,x0): Encuentra una raíz de la función f (x) = fun, que debe ser definida antes o allí mismo con la instrucción *inline*. Esta instrucción busca un cero de f cerca del punto x0 especificado por el usuario, y se basa en los métodos de la secante, bisección e interpolación cuadrática inversa, por lo que la función ingresada debe ser continua.
Por ejemplo, podemos digitar:
>> x=fzero(inline('0.5*exp(x/3)-sin(x)'),0)
con la cual obtenemos: x = 6.7721e-001.

feval: sirve para evaluar, dentro de una función, otra función cuyo nombre se ha recibido como argumento. Por ejemplo, si dentro de una función se quiere evaluar la función ***calcular(A, b, c)***, donde el nombre ***calcular*** se envía como argumento en la cadena ***nombre***, entonces ***feval(nombre, A, b, c)*** equivale a ***calcular(A, b, c)***.

fsolve: Resuelve funciones no lineales en forma simbólica F(x)=0

3.2 Gráficas con MATLAB

Gráficas 2D
Funciones de la forma y = *f(x)*
Dibujar la grafica de la función *y=sen(x)*

```
>>x=0:pi/100:2*pi;
>>x=linspace(0,2*pi,200);
>> y = sin(x);
>>plot(x,y)
```

Curvas en Paramétricas
Dibujar la gráfica de la siguiente curva

$$\vec{r}(t)=\left(\frac{t(t^2-1)}{t^2+1},\frac{2(t^2-1)}{t^2+1}\right); \qquad -5\le t\le 5$$

```
>>t=linspace(-5,5,1000);
>>plot((t.*(t.^2-1))./(t.^2+1),(2*(t.^2-1))./(t.^2+1))
```

Gráficas 3D
Curvas en el espacio
Dibujar la hélice:
$\vec{r}(t)=(sen(t),\cos(t),t); \qquad 0\le t\le 8\pi$

```
>>t=linspace(0,8*pi,2000);
>>plot3(sin(t),cos(t),t),grid on
```

Funciones de la forma z=f(x,y)
Dibujar la gráfica de la función

$z=e^{-(x^2+y^2)}$

En la región del plano $D=\{(x,y)/-2\le x\le 2,\ -2\le y\le 2\}$

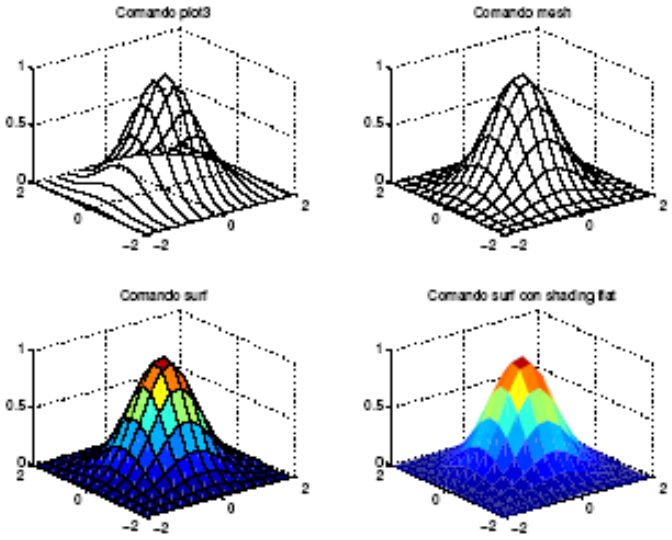
Generamos el mallado

```
>>[x,y]=meshgrid(-2:.5:2);
```

Sustituimos en la función para calcular los valores de z

```
>>z=exp(-x.^2-y.^2);
```

Y ahora podemos dibujar el gráfico con alguno de los siguientes comandos que producen los dibujos mostrados en la figura:

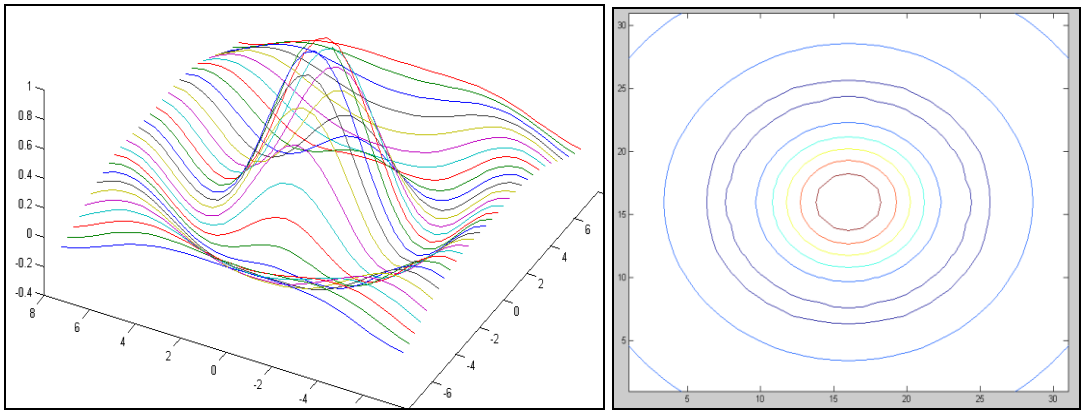



```
>>plot3(x,y,z)
>>mesh(x,y,z)
>>surf(x,y,z)
>>surf(x,y,z)
```

Curvas de Nivel

El siguiente gráfico muestra las curvas de nivel para la siguiente superficie.

```
[X,Y]=meshgrid(-7.5:0.5:7.5);
Z = sin(sqrt(X.^2+Y.^2))./ sqrt(X.^2+Y.^2);
Surf(X,Y,Z)
contour(Z)
```



4. Parte Práctica

Ejemplo 1: Bisección

```
» f=inline('exp(-x)-x')
```

```
f =
    Inline function:
    f(x) = exp(-x)-x
» format long
» fzero(f,1)
Zero found in the interval: [0.54745, 1.32].
ans =
  0.56714329040978
```

```
% bolzano.m
function [raiz,z,it]=bolzano(f,a,b,TOL)
z=[];
for it=1:1000
  x=(a+b)/2;
  err=(b-a)/2;
  z=[z; a x b err];
  if feval(f,a)*feval(f,x)<0
    b=x;
  else
    a=x;
  end
  if err<TOL
    break
  end
end
raiz=x;
```

Por ejemplo se desea hallar la raíz comprendida entre 0 y 1 con una precisión de 1e-4. z contiene los resultados parciales: a, x, b y err.

```
» [raiz,z,it]=bolzano(f,0,1,1e-4)
raiz =   0.56719970703125
z =
  0                  0.50000000000000  1.00000000000000  0.50000000000000
  0.50000000000000  0.75000000000000  1.00000000000000  0.25000000000000
  0.50000000000000  0.62500000000000  0.75000000000000  0.12500000000000
  0.50000000000000  0.56250000000000  0.62500000000000  0.06250000000000
  0.56250000000000  0.59375000000000  0.62500000000000  0.03125000000000
  0.56250000000000  0.57812500000000  0.59375000000000  0.01562500000000
  0.56250000000000  0.57031250000000  0.57812500000000  0.00781250000000
  0.56250000000000  0.56640625000000  0.57031250000000  0.00390625000000
  0.56640625000000  0.56835937500000  0.57031250000000  0.00195312500000
```

0.56640625000000	0.56738281250000	0.56835937500000	0.00097656250000
0.56640625000000	0.56689453125000	0.56738281250000	0.00048828125000
0.56689453125000	0.56713867187500	0.56738281250000	0.00024414062500
0.56713867187500	0.56726074218750	0.56738281250000	0.00012207031250
0.56713867187500	0.56719970703125	0.56726074218750	0.00006103515625

it =

14

Ejemplo 2 Localización

a) Localizar las raíces de la función $f(x)=\frac{e^{x/3}}{2}-sen(x)$ en el intervalo [-10,10]

Creamos la siguiente función:

```
% fun1.m
function [f]=fun1(x)
  f=1/2*exp(x/3)-sin(x);
```

Sea el siguiente programa para localizar las raíces gráficamente y mostrara los intervalos unitarios que contengan raíces:

```
% Localiza.m
clc, clear all, format short
x=-10:10;
y=fun1(x);
plot(x,y),grid
disp('x vs y')
disp([x' y'])
% Intervalos que contienen raices
acu=[];
for i=1:length(x)-1
  if y(i)*y(i+1)<0, acu=[acu; x(i) x(i+1)];
      end
end
disp('Intervalos que contienen raices...'); disp(acu)
```

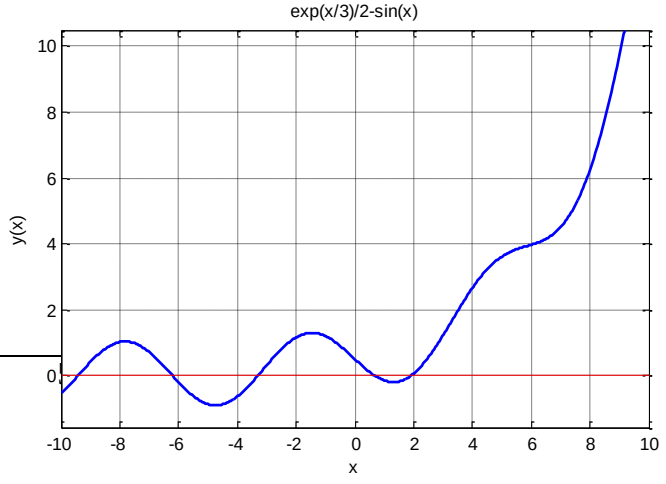
Corrida del Programa

» localiza

Corrida del Programa

x vs y

-10.0000	-0.5262
-9.0000	0.4370
-8.0000	1.0241
-7.0000	0.7055
-6.0000	-0.2117
-5.0000	-0.8645
-4.0000	-0.6250
-3.0000	0.3251



```
  -2.0000    1.1660
  -1.0000    1.1997
        0    0.5000
   1.0000   -0.1437
   2.0000    0.0646
   3.0000    1.2180
   4.0000    2.6536
   5.0000    3.6062
   6.0000    3.9739
   7.0000    4.4991
   8.0000    6.2066
   9.0000    9.6306
  10.0000   14.5598
Intervalos que contienen raíces...
  -10    -9
   -7    -6
   -4    -3
    0     1
    1     2
```

b) Revuélvase la ecuación anterior usando el método de Newton-Raphson con una Tolerancia de 1e-8:

```
% dfun1.m
function [df]=dfun1(x)
   df=1/6*exp(x/3)-cos(x);
```

```
% raphson.m
function [acu,raiz,it]=raphson(f,df,x,TOL)
acu=[];
for it=1:100
   xn=x-feval(f,x)/feval(df,x);
   err=abs(xn-x);
   acu=[acu; xn err];
   x=xn;
   if err<TOL
      break
   end
end
raiz=xn;
```

Corrida del Programa

```
» [acu,raiz,it]=raphson('f','df',1.5,1e-8)

acu =
   2.34849118108965   0.84849118108965
   1.99088685483220   0.35760432625745
   1.91178545812247   0.07910139670974
   1.90682391153077   0.00496154659170
   1.90680389529660   0.00002001623416
```

```
   1.90680389497052   0.00000000032609
raiz =
   1.90680389497052
it =
     6
```

Se observa una convergencia bastante rápida.

Nota.- Si las funciones son almacenas en archivo su nombre debe ir entre apostrofes, si son funciones inline de memoria no deben llevar apostrofes.

```
» f=inline('1/2*exp(x/3)-sin(x)')
» df=inline('1/6*exp(x/3)-cos(x)');
» [acu,raiz,it]=raphson(f,df,1.5,1e-8)
```

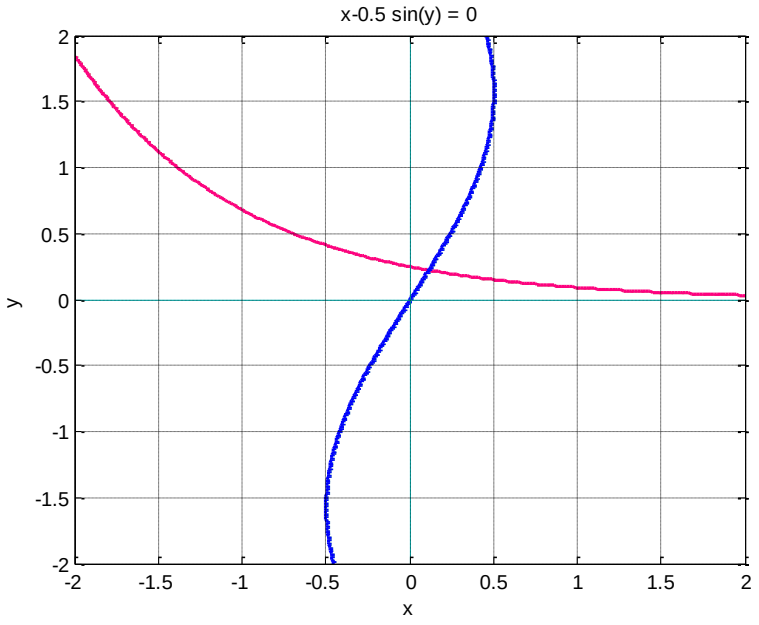
c) Implementar una rutina en MATLAB para el método de aproximaciones sucesivas.

Ejemplo 3 de solución de Sistema No Lineal

$$\begin{cases} x1 - \frac{1}{2}\sin(x2) = 0 \\ x_2 - \frac{1}{4}\exp(-x_1) = 0 \end{cases}$$

Localización de la raíces
En Matlab usar los comandos:

```
f=@(x,y) x-0.5*sin(y);
f1=@(x,y) y-0.25*exp(-x);
ezcontour(f,[-2 2],50)
hold on
ezcontour(f1,[-2 2],50)
hold on
ezcontour(f1,[-2 2],50)
```



Sólo una raíz cercana a $[0\ \ 0]^t$

Resolver usando:

Método del punto fijo

Arreglando el sistema no lineal en un equivalente

$$\begin{aligned} x_1 &= \frac{1}{2}\sin(x_2) \\ x_2 &= \frac{1}{4}\exp(-x_1) \end{aligned}$$

$$x = G(x)$$

Probar si es algoritmo convergente
Obetener el jacobiano de G

$$G'(x) = \begin{bmatrix} 0 & \frac{1}{2}\cos(x_2) \\ -\frac{1}{4}\exp(-x_1) & 0 \end{bmatrix}$$

En Matlab usar los comandos:

```
syms x1 x2
G=[x1-0.5*sin(x2); x2-0.25*exp(-x1)]
dG=jacobian(G,[x1 x2])
```

$\|G'\|_\infty = \frac{1}{2} < 1$

```
dG=subs(dG,{x1,x2},{0  0})
norm(dG,inf)
```

Por lo tanto converge
Algoritmo

$$\boxed{\begin{aligned} x_1^{(k+1)} &= \frac{1}{2}\sin(x_2^{(k)}) \\ x_2^{(k+1)} &= \frac{1}{4}\exp(-x_1^{(k)}) \end{aligned}}$$

Condiciones iniciales: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Tolerancia= $0.5*10^{-3}$

Max_it=50

Criterio de Parada: $e_k = \left\| x^{(k+1)} - x^{(k)} \right\|_\infty$

Iteraciones:

i	x_1	x_2	error	
0	0.00000000	0.00000000	1.00000000e+000	
1	0.00000000	0.25000000	2.50000000e-001	
2	0.12370198	0.25000000	1.23701980e-001	
3	0.12370198	0.22091079	2.90892137e-002	
4	0.10955918	0.22091079	1.41427964e-002	
5	0.10955918	0.22405728	3.14649393e-003	
6	0.11109365	0.22405728	1.53446949e-003	
7	**0.11109365**	**0.22371373**	**3.43545414e-004**	**(cumple)**

```
function[H]=pto_fs(G,x0,tol,maxit)
syms x1 x2
H=[0 x0' 1];
for i=1:maxit
 x=subs(G,{x1 x2},{x0(1) x0(2)});
 e=norm(x-x0,inf);
 H=[H; i,x' e];
 if e<tol
     break
 end
 x0=x;
end
```

Importante:
Grabar como pto_fs.m
Corrida: usar un programa llamado prob_sist.m con el siguiente código:

```
clear all
syms x1 x2
G=[0.5*sin(x2); 0.25*exp(-x1)]
x0=[0 0]';
maxit=50;
tol=0.5e-3
H=pto_fs(G,x0,tol,maxit);
disp('   i          x1             x2                  error')
fprintf('%4.0f %12.8f %16.8f %20.8e \n',H')
```

Ejemplo 4 Newton Raphson para Sistemas

Resolver el siguiente sistema usando el método de Newton Raphson:

$$y + x^2 - 0.5 - x = 0$$

$$x^2 - 5xy - y = 0$$

Valor inicial : $x = 1,\ \ y = 0.5$

Solución

$$F=\begin{bmatrix} y+x^2-0.5-x \\ x^2-5xy-y \end{bmatrix},\ F'=\begin{bmatrix} 2x-1 & 1 \\ 2x-5y & -5x-1 \end{bmatrix},\ X_0=\begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

Iteración 1:

$$F=\begin{bmatrix} y+x^2-0.5-x \\ x^2-5xy-y \end{bmatrix}=\begin{bmatrix} -0.5 \\ 1 \end{bmatrix}=,\ F'=\begin{bmatrix} 2x-1 & 1 \\ 2x-5y & -5x-1 \end{bmatrix}=\begin{bmatrix} 1 & 1 \\ 2 & -6 \end{bmatrix}$$

$$X_1=\begin{bmatrix} 1 \\ 0 \end{bmatrix}-\begin{bmatrix} 1 & 1 \\ 2 & -6 \end{bmatrix}^{-1}\begin{bmatrix} -0.5 \\ 1 \end{bmatrix}=\begin{bmatrix} 1.25 \\ 0.25 \end{bmatrix}$$

Iteración 2:

$$F=\begin{bmatrix} 0.0625 \\ -0.25 \end{bmatrix}=,\ F'=\begin{bmatrix} 1.5 & 1 \\ 1.25 & -7.25 \end{bmatrix}$$

$$X_2=\begin{bmatrix} 1.25 \\ 0.25 \end{bmatrix}-\begin{bmatrix} 1.5 & 1 \\ 1.25 & -7.25 \end{bmatrix}^{-1}\begin{bmatrix} 0.0625 \\ -0.25 \end{bmatrix}=\begin{bmatrix} 1.2332 \\ 0.2126 \end{bmatrix}$$

En dos iteraciones presenta una aproximación de …E=……..

Ejemplo 5

Aproximar la solución al siguiente sistema de ecuaciones:

$$\begin{cases} x^3-xy^2+y^3=0 \\ x\sin(xy)+1=0 \end{cases}$$

Tenemos con $\bar{x}=(x,y)^T$ que

$$\bar{F}(x)=\begin{bmatrix} x^3-xy^2+y^3 \\ x\sin(xy)+1 \end{bmatrix},\qquad J_F(x)=\begin{bmatrix} 3x^2-y^2 & -2xy+3y^2 \\ \sin(xy)+xy\cos(xy) & x^2\cos(xy) \end{bmatrix}$$

Estas dos expresiones las calculamos en MATLAB mediante las siguientes funciones

```
function z=func1(w)
z=zeros(2,1);
x=w(1);
y=w(2);
z(1)=x^3-x*y^2+y^3;
z(2)=x*sin(x*y)+1;
```

```
function z=dfunc1(w)
z=zeros(2,2);
x=w(1);
y=w(2);
z(1,1)=3*x^2-y^2;
z(1,2)=-2*x*y+3*y^2;
z(2,1)=sin(x*y)+x*y*cos(x*y);
z(2,2)=x^2*cos(x*y);
```

```
function [x,iter]=newton(f,fp,x0,tol,itermax)
%NEWTON Método de Newton para sistemas no lineales
% Los datos de entrada son
% f: Nombre de la función que representa el sistema.
% fp: Nombre de la función que calcula el Jacobiano.
% x0: El punto inicial (vector columna).
% tol: Tolerancia para el error relativo  en la solución calculada
% itermax: Número máximo de iteraciones que se repiten las
iteraciones
if nargin<4
   tol=1.0e-4;
end
if nargin<5
   itermax=20;
end
x=x0;
normx=0;
normz=inf;
iter=0;
while (normz>tol*normx)&(iter<=itermax)
   f0=feval(f,x);
   fp0=feval(fp,x);
   z=-fp0\f0;
   normz=norm(z,2);
   normx=norm(x,2);
   x=x+z;
   iter=iter+1;
end
```

Esta función se debe invocar con al menos tres argumentos. Si se omite alguno de los últimos dos argumentos, la función tiene unos valores que se asignan por omisión a estas variables.

Tomando como $\bar{x} = (1,0)^T$, podemos invocar la función newton para resolver el sistema como sigue:

[x,iter]=newton('func1','dfunc1',[1,0])

Ejemplo 6

Para el siguiente sistema no lineal

$$\begin{aligned} 2x_1 - \sin(x_2) &= 0 \\ 4x_2 - \exp(-x_1) &= 0 \end{aligned}$$

Definir F(x) y el Jacobiano de F(x)

$$F(x) = \begin{bmatrix} 2x_1 - \sin(x_2) \\ 4x_2 - \exp(-x_1) \end{bmatrix} \qquad \text{für } x = (x_1, x_2)$$

$$F'(x) = \begin{bmatrix} 2 & -\cos(x_2) \\ \exp(-x_1) & 4 \end{bmatrix}$$

En Matlab:

```
function [y,J] = F(x)
y = [  2*x(1) - sin(x(2)) ; 4*x(2) - exp(-x(1)) ];
J = [ 2 , -cos(x(2)) ; exp(-x(1)) , 4 ];
```

Para hacer un paso de Newton usar la función

```
function y = newtonstep(x)
[y,J] = F(x);
dx = - J \ y;
y = dx + x;
```

Para aplicar Newton al problema

```
clear all
x0=[0 0]';
maxit=50;
tol=0.5e-3;
H=[0 x0' 1];
x=x0;
for i=1:maxit
    y = newtonstep(x);
    e=norm(y-x,inf);
    H=[H; i y' e];
    if e<tol
        break
    end
    x=y;
end
disp('   i          x1             x2
error')
fprintf('%4.0f %12.8f %16.8f %20.8e \n',H')
```

```
>> aplica_newton
   i      x1               x2                 error
   0   0.00000000       0.00000000      1.00000000e+000
   1   0.11111111       0.22222222      2.22222222e-001
   2   0.11094275       0.22374749      1.52527019e-003
   3   0.11094264       0.22374752      1.14444473e-007
```

FUNCIONES IMPORTANTES

function biseccion(f,a,b,tol)

```
f=inline(f);
% inline convierte a f en una función que depende de x
n=ceil(log((b-a)/tol)/log(2));
% ceil toma el entero mayor cercano obtenido por la cota de error del método
fprintf('\n it. a b x f(x) \n')
for i=1:n
    x=(a+b)/2;
    fprintf('%3.0f %10.10f %10.10f %10.10f %10.10f \n',i,a,b,x,f(x))
    % muestra en cada paso los valores de la iteración, de a, de b, de x y de
    f(x)
    % la instrucción %10.10f significa dejar 10 espacios y colocar el número
    con 10 decimales
    % la instrucción \n se emplea para cambiar a línea nueva
    if f(a)*f(x)<0
        b=x;
        else
        a=x;
    end
end
fprintf('\n La aproximación de la raíz es: %3.10f \n\n',x)
```

function newton(f,x0,tol)

```
sym x;
% convierte a x en una variable simbólica para poder derivar la función
df=diff(f,'x');
% deriva con respecto a x
f=inline(f);
df=inline(char(df));
% inline convierte a f y su derivada df en una funciones que dependen de x
% char transforma a la derivada como una cadena de caracteres para poder
% definirla como función
fprintf('\n it. x f(x) \n')
i=0;
fprintf('%3.0f %10.10f %10.10f \n',i,x0,f(x0))
x1=x0-f(x0)/df(x0);
while abs(x0-x1)>tol
    i=i+1;
    % contador de iteraciones
    fprintf('%3.0f %10.10f %10.10f \n',i,x1,f(x1))
    x0=x1;
    x1=x0-f(x0)/df(x0);
end
fprintf('\n La aproximación de la raíz es: %3.10f \n\n',x1)
function puntofijo(g,x0,tol)
g=inline(g);
fprintf('\n it. x g(x) \n')
i=0;
```

```
fprintf('%3.0f %10.10f %10.10f \n',i,x0,g(x0))
x1=g(x0);
while abs(x0-x1)>tol
        i=i+1;
        fprintf('%3.0f %10.10f %10.10f \n',i,x1,g(x1))
        24
        x0=x1;
        x1=g(x0);
end
fprintf('\n La aproximación del punto fijo es %3.10f \n\n',x1)
```

function newtonnl(f1,f2,X0,tol,mx)

```
syms x y;
j=jacobian([f1;f2],[x y]);
% obtiene la matriz jacobiana para f1 y f2 en las variables x y y
F1=inline(char(f1),'x','y');
% los paremetros 'x','y' permiten definir la función en terminos de dos variables
F2=inline(char(f2),'x','y');
J1=inline(char(j(1,1)),'x','y');
J2=inline(char(j(1,2)),'x','y');
J3=inline(char(j(2,1)),'x','y');
J4=inline(char(j(2,2)),'x','y');
% define cada función dependiente de las variables x y y
E=tol+1;
% inicializamos una variable para medir la distancia entre dos aproximaciones
% como tol+1 para que el cilo empiece
C=0;
% nos permite contar las iteraciones que vamos realizando de modo que no
% sobrecen a mx que es el número máximo de iteraciones.
F=zeros(2,1); J=zeros(2,2);
% inicializa el vector columna para que el producto este bien definido
% e inicializa la matriz jacobiana en ceros
while E>tol & C<mx
        C=C+1; % cuenta la iteración
        F(1)=F1(X0(1),X0(2)); F(2)=F2(X0(1),X0(2));
        fprintf('n=%1.0f x=%12.10f y=%12.10f ',C,X0(1),X0(2));
        25
        %muestra paso de la iteracion, la aproximación para x y y
        fprintf('f1(x,y)=%12.5e f2(x,y)=%12.5e \n',F(1),F(2))
        %muestra el copmportamiento de las funciones del sistema (esperamos
        sean casi cero)
        J(1,1)=J1(X0(1),X0(2)); J(1,2)=J2(X0(1),X0(2));
        J(2,1)=J3(X0(1),X0(2)); J(2,2)=J4(X0(1),X0(2));
        H=-J\F;
        X1=X0+H;
        E=norm(X1-X0);
        X0=X1;
end
```

5. Ejercicios Propuestos

1. Crear un archivo *enlfx.m* (ecuaciones no lineales *f*(*x*)) que contendrá la función cuyas raíces se quieren hallar. En particular, tomaremos $f(x)=2x\text{Cos}(2x)-(x+1)^2$. Para representar la función en el intervalo [-6, 6], se puede utilizar:
   » z=linspace(-6,6,50);fz=enlfx(z);
   » plot(z,fz);grid on;

2. Use cuatro iteraciones del método del la Bisección para determinar las raíces de $e^{2x}-6x=0$ en el intervalo [0,0.5]. ¿Cuántas iteraciones son necesarias para obtener la aproximación a la raiz redondeada a 5 cifras decimales?
   a=0, b=0.5 f(a)= 1 f(b)= -0.2817 f(a)f(b)<0

Iteration	a	b	x	f(x)	Error
0	0	0.5000	0.2500	0.1487	0.25
1					
2					
3					
4					

   No de iteraciones para obtener 5 c.d.e.=……..

3. Desarrolle cuatro iteraciones usando el método de Newton Raphson para obtener las raíces de $e^{2x}-6x=0$. Use x= 0.4 como valor inicial. Dar un estimado para el error involucrado.

Iteración	x	f(x)	f'(x)	f(x)/f'(x)
0				
1				
2				
3				
4				

4. Sea la función no lineal $f(x)=3\cos(2x)-x^2$
   ¿Es posible encontrar un algoritmo del punto fijo en $x\in[0,1]$? Justifique.
   Si su respuesta es afirmativa realice 03 iteraciones.
   ……
   ……
   ……
   ……
   ……

5. Programar la función pfijo.m según el algoritmo siguiente, usando como función de iteración $g(x)=\sqrt{\frac{x+3}{x^2+2}}$ para encontrar la aproximación con un error menor que 0.001 tomando un valor inicial aleatorio en el intervalo [1,2]. Escribir los resultados en la tabla, indicando el número de iteraciones 'n'.

   Entrada: nombre del archivo que contiene la función g(x), valor inicial 'z', error admisible 'ex' de la raíz y número máximo de iteraciones 'niter'.
   Salida: vector x de sucesiones de aproximaciones a la solución.
   Paso 1 Verificar los argumentos de entrada (mínimo 2), tomando por defecto ex=eps,
   niter=1000
   Paso 2 Definir xniterx1
   Paso 3 Hacer x1=z
   Paso 4 Repetir para k desde 2 hasta niter los Pasos 5-6
   Paso 5 Hacer xk=g(xk-1);
   Paso 6 SI $|x_k-x_{k-1}|<$ex FINALIZAR BUCLE
   Paso 7 SI k>niter, MENSAJE 'Finde iteraciones sin convergencia'
   Paso 8 SINO eliminar elementos $x_{k+1}, x_{k+2}, \ldots x_{niter}$

   Para la ejecución del programa es necesario generar previamente la función de punto fijo de Newton-Raphson, cuyo código, aprovechando que es un polinomio, se da a continuación.

```
function y=gx(x)
% Definición de g(x)
p=[1,0,2,-1,-3]; px=polyval(p,x);
q=polyder(p);qx=polyval(q,x);
y=x-px./qx;
```

6. Se desea resolver x^2-sen(x)=0 usando Newton-Raphson mediante un programa en MATLAB, complete las instrucciones que faltan:

```
x=1                 % aproximación inicial
tol=…… % precisión de 6 cifras decimales exactos
err=1
while err>tol

xn=……

err=abs(xn-x)

…….....
end
```

7. Dado el siguiente sistema no lineal:

$$\begin{cases} x^2+y^2=1 \\ xy+x=1 \end{cases}$$

   Localizar gráficamente la solución del sistema. Dado el punto inicial $(x^{(0)},y^{(0)})=(1,1)$ aproximar la solución del sistema usando 02 iteraciones de las

aproximaciones sucesivas. Use comandos del MATLAB para obtener la solución exacta.

.....
.....
.....
.....
.....

8. Dado el siguiente sistema no lineal resuelva utilizando el método de Aproximaciones sucesivas o Newton-Raphson

$$y + x^2 - 1 - x = 0$$
$$x^2 - 2y^2 - y = 0$$
Condición inicial $x = 0,\ y = 0$

9. Desarrolle funciones para calcular la solución de un sistema de ecuaciones no lineales, aplicando el método de Newton-Raphson y Punto Fijo. Considere un parámetro para el error máximo y otro para el número de iteraciones máximo.

10. Resuelva por el método de Newton para sistemas de no lineales, el siguiente sistema

$$\begin{cases} f_1(x,y) = 1 + x^2 - y^2 + e^x \cos(y) \\ f_2(x,y) = 2xy + e^x \sin(y) \end{cases}$$

Use valores iniciales $x_0 = -1$ y $y_0 = 4$.

11. Comenzando en (0, 0, 1), resuelva por el método de Newton para sistemas no lineales con el sistema

$$\begin{cases} xy - z^2 = 1 \\ xyz - x^2 + y^2 = 2 \\ e^x - e^y + z = 3 \end{cases}$$

explique los resultados.

12. Use el método de Newton para encontrar una raíz del sistema no lineal

$$\begin{cases} 4y^2 + 4y + 52x = 19 \\ 169x^2 + 3y^2 + 111x - 10y = 10 \end{cases}$$

13. Use el método de Newton para calcular la única raíz de:

$$x + e^{-Bx^2}\cos(x) = 0$$

Use una variedad de valores de B, por ejemplo B = 1, 5, 10, 25, 50. Entre las elecciones del punto de partida tomo x0 = 0 y explique el comportamiento anómalo. Teóricamente, el método de Newton debe converger para cualquier valor de x0 y B.

	Curso	**Métodos Numéricos**	**Código : MB536**
	Tema	**Aproximación de Funciones**	
	Practica	**06**	
	Profesores	**Garrido Juárez, Rosa** **Castro Salguero, Robert** **Pantoja Carhuavilca, Hermes** **Obregón Ramos, Máximo** **Ruiz Lizama, Edgar**	

1. Objetivos

Estudiar y aplicar los diversos métodos de aproximación de funciones mediante interpolación polinómica, ajuste de mínimos cuadrados e interpolación por splines.

2. Fundamento Teórico

Problema básico de Interpolación: Dados números distintos, $x_0, x_1, \ldots x_n$ y números arbitrarios $y_0, y_1, \ldots y_n$, queremos hallar una función g(x) tal que

$$g(x_i) = y_i \ , \ 0 \le i \le n.$$

Problema de Interpolación Polinomial: Dados números distintos, $x_0, x_1, \ldots x_n$ y números arbitrarios $y_0, y_1, \ldots y_n$, queremos hallar un polinomio $p_n(x)$ de grado a lo más n, tal que

$$p_n(x_i) = y_i \ , \ 0 \le i \le n.$$

Existencia y construcción de $p_n(x)$

Sea el polinomio de la forma:

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots a_1 x + a_0 \ ,$$

Por condición de interpolación:

$$p_n(x_i) = y_i \quad 0 \le i \le n.$$

Esto es equivalente al sistema:

$$\begin{pmatrix} x_0^n & x_0^{n-1} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_n^n & x_n^{n-1} & \cdots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_2 \end{pmatrix}$$

en el que la matriz del sistema es conocida como *Matriz de Vandermonde*. La existencia y unicidad del polinomio interpolante es *equivalente* a asegurar que el sistema es posible de determinar para cualesquiera x_0 , ... , x_n distintos.

Error de Interpolación:

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n)$$

Diferencias Divididas

Se define para puntos o argumentos desigualmente espaciados:

Diferencia dividida de primer orden:

$$f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

Diferencia dividida de segundo orden:

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$$

Diferencia dividida de orden "n":

$$f[x_i, x_{i+1}, ..., x_{i+n-1}, x_{i+n}] = \frac{f[x_{i+1}, ..., x_{i+n}] - f[x_i, ..., x_{i+n-1}]}{x_{i+n} - x_i}$$

Polinomio de interpolación de Newton basado en diferencias Divididas

$$P_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + f[x_0, x_1, ..., x_n](x - x_0)(x - x_1)...(x - x_{n-1})$$

$$P_n(x) = f(x_0) + \sum_{k=1}^{n} f[x_0, ..., x_k](x - x_0)...(x - x_{k-1}) = f(x_0) + \sum_{i=0}^{n} f[x_0, ..., x_i] \prod_{j=0}^{i-1} (x - x_j)$$

Polinomios de interpolación de Lagrange

Para intervalos igualmente espaciados o no.

$$P_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i) = L_0(x) f(x_0) + L_1(x) f(x_1) + ... + L_n(x) f(x_n)$$

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right)$$

Se cumple:

$$L_j(x_k) = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$$

Ajuste por mínimos cuadrados lineal

Sea el conjunto de datos: $(x_1, y_1); (x_2, y_2); \cdots (x_n, y_n);$

Se pueden ajustar a una recta $y = c_2 x + c_1$ en forma óptima, resolviendo la ecuación normal:

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} c_2 \\ c_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \Rightarrow M\, c = y$$

$M^T M\ c = M^T y \qquad Ec.\ Normal$

$$\begin{bmatrix} \sum x^2 & \sum x \\ \sum x & n \end{bmatrix} \begin{bmatrix} c_2 \\ c_1 \end{bmatrix} = \begin{bmatrix} \sum xy \\ \sum y \end{bmatrix}$$

Este procedimiento se puede extender a polinomios de grado mayor.

Factor de regresión

$$R^2 = \frac{\sum_{i=1}^{n} (\hat{y}_i - y_m)^2}{\sum_{i=1}^{n} (y_i - y_m)^2}$$

$\hat{y}_i$ *de la funcion de ajuste*

y_i *de la data*

$$y_m = \frac{\sum_{i=1}^{n} y_i}{n}$$

Spline cúbico natural

Sea el conjunto de datos: $(x_0, y_0); (x_1, y_1); (x_2, y_2); \cdots (x_n, y_n);$

Donde cada segmento puede ser aproximado con un polinomio cúbico de la forma:

$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad i = 0, 1, \cdots, n-1$

Haciendo: $h_i = x_{i+1} - x_i \quad M_i = S''(x_i)$

Para el spline natural: $M_0 = M_n = 0$

Debemos primero resolver el siguiente sistema tridiagonal:

$$\begin{bmatrix} 2(h_0 + h_1) & h_1 & 0 & \cdots & 0 \\ h_1 & 2(h_1 + h_2) & h_2 & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & h_{n-3} & 2(h_{n-3} + h_{n-2}) & h_{n-2} \\ 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = 6 \begin{bmatrix} f[x_1, x_2] - f[x_0, x_1] \\ f[x_2, x_3] - f[x_1, x_2] \\ \vdots \\ f[x_{n-2}, x_{n-1}] - f[x_{n-3}, x_{n-2}] \\ f[x_{n-1}, x_n] - f[x_{n-2}, x_{n-1}] \end{bmatrix}$$

Una vez obtenidos $M_1, \cdots M_{n-1}$, obtendremos los coeficientes:

$$a_i = \frac{M_{i+1} - M_i}{6h_i}$$

$$b_i = \frac{M_i}{2}$$

$$c_i = f[x_i, x_{i+1}] - \frac{M_{i+1} + 2M_i}{6} h_i$$

$$d_i = y_i$$

3. Instrucciones básicas en MATLAB

1. Representar el siguiente polinomio en MATLAB:
$$P(x) = x^5 + 2x^3 + x^2 - x + 1$$

   » p=[1 0 2 1 -1 1]

2. Evalúe el polinomio anterior en x=1, 2, 4
   » xx=[1 2 4]

   » yy=polyval(p,xx)

3. Obtener la derivada del polinomio anterior:
   » dp=polyder(p)

4. Elabore una función para obtener la integral de un polinomio en el intervalo [a,b], con la cabecera: function I = integ(p,a ,b)

5. Obtener las raíces del polinomio anterior:
   » raices=roots(p)

6. Obtener un polinomio cuyas raíces son: 1, -1, 2, 3
   » r=[1 -1 2 3]

   » p=poly(r)

7. Efectuar: $q(x)=3(x+2)(x-0.5)(x+3)^2(x-1)^3$

8. Multiplicar (x^2-x+1) y (x^3+1)

   » p1=[1 -1 1]

   » p2=[1 0 0 1]

   » prod=conv(p1,p2)

9. Dividir (x^3+4x^2+2x+1) entre (x^2+x+2)

   » p1=[1 4 2 1]

   » p2=[1 1 2]

   » [q,r]=deconv(p1,p2)

10. Escriba una función que permita sumar dos polinomios:
    function s=sumpoly(p1,p2)

11. Obtener el polinomio interpolante que pase por los siguientes puntos:

X	1	2	4	5
Y	3	10	66	127

    » x=[1 2 4 5]'

    » y=[3 10 66 127]'

    » M=[x.^3 x.^2 x ones(4,1)]

    » p=M\y % Coeficientes del polinomio interpolante

12. Aproxime la función para x=3, 4.5

    » xi=[3 4.5]

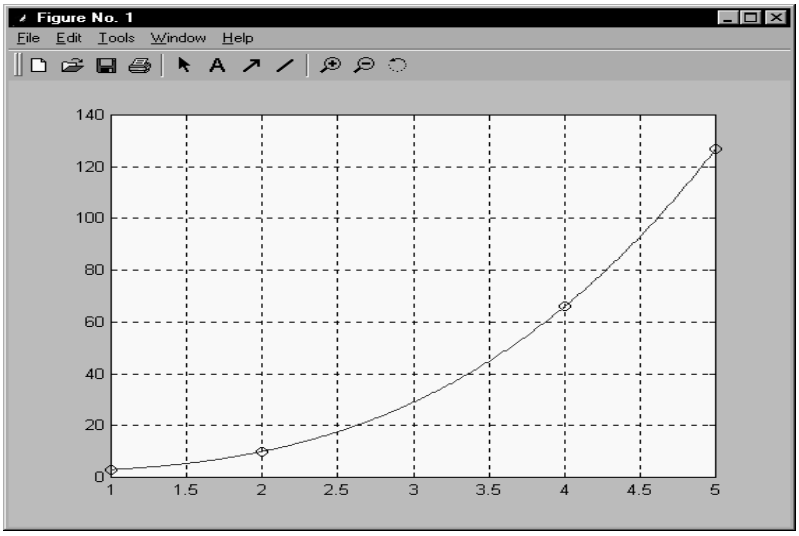
» yi=polyval(p,xi)

13. Graficar el polinomio anterior:

» xx=1:0.01:5;

» yy=polyval(p,xx);

» plot(x,y,'o',xx,yy)



14. Obtenga un polinomio interpolante de cuarto grado que aproxime a f(x)= sin(x) tomando los puntos: x=0, 0.2, 0.3, 0.5 0.7. Aproxime mediante el polinomio f(0.1), f(0.4) y f(0.6) y muestre el error. Grafique f(x) vs $P_4(x)$

15. Mediante el comando ***polyfit*** obtener un polinomio interpolante que pase por los puntos:

X	0	1	3	4	6
Y	1	0	64	225	1225

Aproxime Y(2), Y(5)

Grafique el polinomio interpolante, paso=0.01.

» x=[0 1 3 4 6]

» y=[1 0 64 225 1225]
» p=polyfit(x,y,length(x)-1)

» xi=[2 5]

» yi=polyval(p,[2 5])

» xx=0:0.1:6;

» yy=polyval(p,xx);

» plot(x,y,'o',xi,yi,'x',xx,yy)

» legend('data','estimados','polinomio')

16. Obtenga un polinomio interpolante de cuarto grado que aproxime la función **$f(x)= e^{-x}$ sen(x)** tomando los puntos: x=0, 0.2, 0.3, 0.5 0.7. Aproxime mediante el polinomio: f(0.1), f(0.4) y f(0.6) y muestre el error. Grafique f(x) vs $P_4(x)$

17. Interpolar sen(x) en el intervalo [0,pi] mediante polinomios, tomando 3, 4 y 5 puntos igualmente espaciados. ¿Cuál es la aproximación más adecuada?

18. Elabore una rutina para construir una tabla de diferencias divididas y permita realizar una interpolación mediante el polinomio de Newton.

La función *difdivididas* implementa el **método de interpolación de Newton usando las Diferencias Divididas.**

```
function z=difdivididas(x,y)
n=length(x);
for j=1:n
  v(j,1)=y(j);
end

fprintf('Diferencia Divididas:\n\n');

for i=2:n
  for j=1:n+1-i
     v(j,i)=(v(j+1,i-1)-v(j,i-1))/(x(j+i-1)-x(j));
     fprintf('  %10.4f',v(j,i));
  end
fprintf('\n\n');
end
for i=1:n
  c(i)=v(1,i);
end

p=[y(1)];
for j=2:n;
   q=poly(x([1:j-1]));
   p=[0,p]+c(j)*q ;
end
fprintf('El polinomio de Newton es:\n');
z=p;
```

Ejemplo.-

%Interpolar por Newton con los siguientes puntos

```
>> x = [0 0.5 1]
>> y = [1 0.8 0.5]
x =
         0    0.5000    1.0000
y =
    1.0000    0.8000    0.5000

>> difdivididas(x,y)
Diferencia Divididas:
     -0.4000     -0.6000
      -0.2000
```

```
El polinomio de Newton es:

ans =

 -0.2000  -0.3000   1.0000   % Polinomio Interpolante de Grado 2
                             % en forma descendente
```

19. Elabore un programa que realice la interpolación de Lagrange:

La función *Lagrange* implementa el **Método de Interpolación de Lagrange**

```
function p = lagrange(x,y)
 n = length(x);
 p = zeros(1,n);
   for j=1:n
      q = poly(x([1:(j-1), (j+1):n]));
      L=q/polyval(q,x(j));
    p = p +  L*y(j);
end
```

20. Realice una interpolación lineal para los siguientes puntos:

» x=[0 0.25 0.5 0.75 1]'

» y=[0.9162 0.8109 0.6931 0.5596 0.4055]'

» xi=[0.2 0.4 0.6]'

» yi=interp1(x,y,xi,'linear')

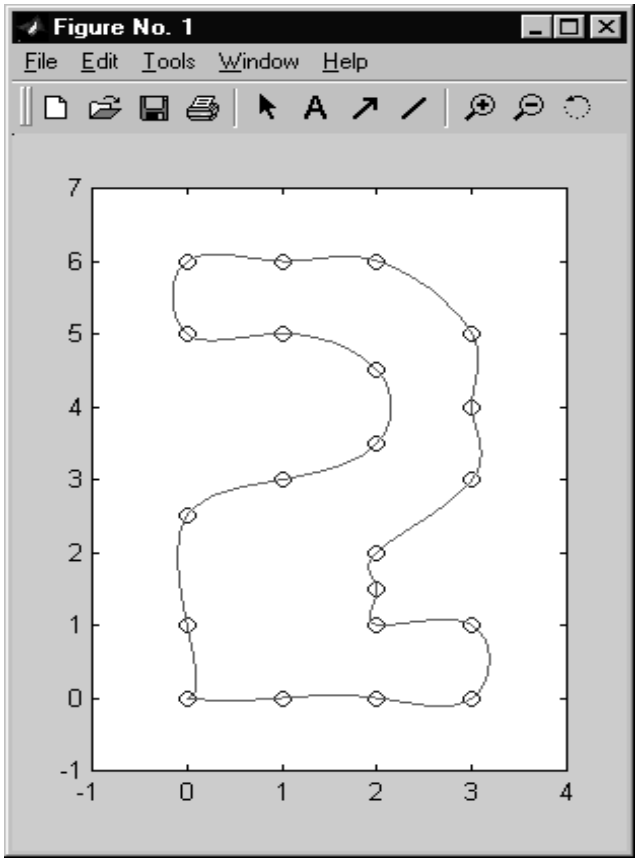
» plot(x,y,'o',xi,yi,'x')

21. Realice una interpolación por splines, para:

```
% ejmspl.m
x=[0.9 1.3 1.9 2.1 2.6 3 3.9 4.4 4.7 5 6];
y=[1.3 1.5 1.85 2.1 2.6 2.7 2.4 2.15 2.05 2.1 2.25];
xx=0.9:0.01:6;
yy=spline(x,y,xx);
plot(x,y,'o',xx,yy)
legend('data','spline')
```

22. Construya una figura uniendo puntos mediante Splines:

```
% pruespl.m
x=[0 1 2 3 3 2 2 2 3 3 3 2 1 0 0  1 2 2 1 0 0 0]
y=[0 0 0 0 1 1 1.5 2 3 4 5 6 6 6 5 5 4.5 3.5 3 2.5 1 0]
p=1:length(x);
pp=1:0.01:length(x);
xx=spline(p,x,pp);
yy=spline(p,y,pp);
plot(x,y,'o',xx,yy)
```



23. Ajustar los siguientes datos a una recta:

X	0.1	0.4	0.5	0.7	0.7	0.9
Y	0.61	0.92	0.99	1.52	1.47	2.03

Se ajusta a una recta: $y = c_1 x + c_2$

Se plantea el sistema: $\begin{bmatrix} 0.1 & 1 \\ 0.4 & 1 \\ 0.5 & 1 \\ 0.7 & 1 \\ 0.7 & 1 \\ 0.9 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0.61 \\ 0.92 \\ 0.99 \\ 1.52 \\ 1.47 \\ 2.03 \end{bmatrix}$

Este sistema: M c = y

Multiplicando por M^t: $M^t M c = M^t y$

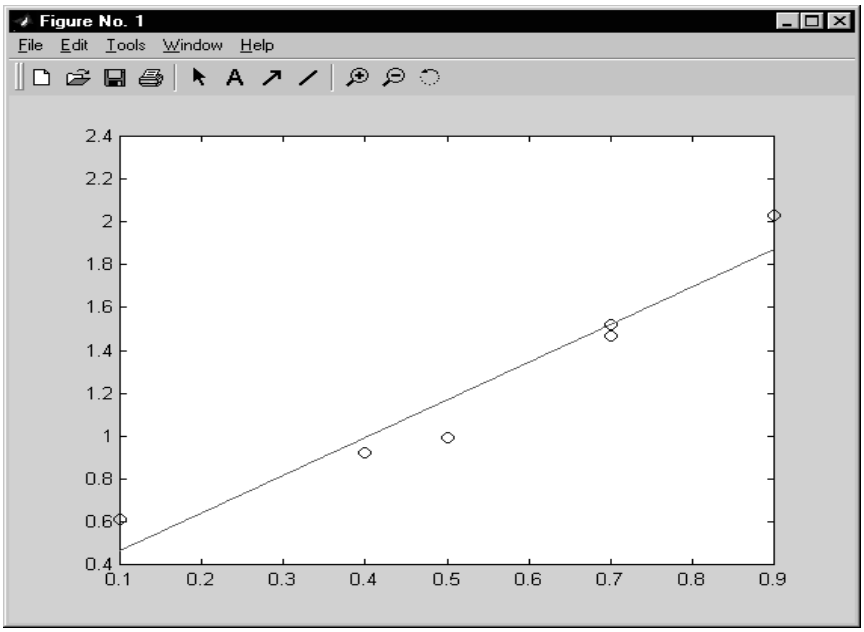
Donde $M^t M$ es una matriz cuadrada y se puede resolver para "c":

```
» x=[0.1 0.4 0.5 0.7 0.7 0.9]'
» y=[0.61 0.92 0.99 1.52 1.47 2.03]'
» M=[x ones(6,1)]
» A=M'*M
```

```
A =
   2.2100   3.3000
   3.3000   6.0000

» b=M'*y

b =
   4.8440
   7.5400

» c=A\b

c =
   1.7646
   0.2862
```

También se puede obtener directamente:
» c=(M'*M)\(M'*y)
» c=M\y
» c=polyfit(x,y,1)

24. Graficar los datos vs la recta de ajuste:



25. Repita el procedimiento para realizar un ajuste cuadrático: $y=c_1x^2+c_2x+c_3$

26. Obtener el factor de regresión: R^2

27. Dada la siguiente función:

$$y = \frac{1 + x}{1 + 2x + 3x^2}$$

Tabule para los puntos: x = 0, 2, 4, 6, 8, 10

Muestre en un sólo gráfico:

a) Un polinomio de grado 5 que pase por todos los puntos
b) Una ajuste lineal
c) Un ajuste cuadrático
d) Una función spline
e) La función exacta

4. Parte práctica

Problema 1

Se desea estimar la densidad de una sustancia a una temperatura de 251º C a partir de los siguientes datos experimentales que se dan en la siguiente tabla.

Tabla **Datos de Temperatura-Densidad**

i	0	1	2
$T_i, ^oC$	94	205	371
$\rho_i, \frac{kg}{m^3}$	929	902	860

Como se dispone de tres datos, el orden de la fórmula de Lagrange es 2 y el cálculo de la densidad a 251 es dado por

$$\begin{aligned}\rho(251^oC) &= \frac{(251-205)(251-371)}{(94-205)(94-371)}(929) \\ &+ \frac{(251-94)(251-371)}{205-94(205-371)}(902) \\ &+ \frac{(251-94)(251-205)}{(371-94)(371-205)}(860) \\ &= 890.5\ kg/m^3\end{aligned}$$

El siguiente procedimiento codificado con MATLAB realiza los cálculos anteriores.

Procedimiento

```
X = [94 205 371];
Y = [929 902 860];
Xi= 251;
Densidad =interp1(X,Y,Xi,'cubic')
```

Problema 2

Obtener una interpolación por **Spline Natural** para el polinomio $p(x) = x^4$, para x=0, 1, 2 y 3.

a) Muestre las funciones Spline *S(x)* para cada intervalo.
b) Dime el máximo error cometido, para ello tabule $|p(x)-S(x)|$ con $\Delta x = 1/4$.
c) Comente sus resultados.

Solución

a)

i	hi	x	F(x)	f[,]
0	1	0	0	1
1	1	1	1	15
2	1	2	16	65
		3	81	

En este caso:

$$\begin{bmatrix} 2(h_0+h_1) & h_1 \\ h_1 & 2(h_1+h_2) \end{bmatrix}\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = 6\begin{bmatrix} f[x_1x_2]-f[x_0x_1] \\ f[x_2x_3]-f[x_1x_2] \end{bmatrix}$$

Reemplazando:

$$\begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = 6\begin{bmatrix} 15-1 \\ 65-15 \end{bmatrix} = \begin{bmatrix} 84 \\ 300 \end{bmatrix}$$

$$M_1 = 2.4 \quad M_2 = 74.4 \quad M_0 = M_3 = 0$$

$$S(x) = \begin{cases} x \in [0,\ 1] & 0.4(x-0)^3 + 0(x-0)^2 + 0.6(x-0) + 0 \\ x \in [1,\ 2] & 12(x-1)^3 + 1.2(x-1)^2 + 1.8(x-1) + 1 \\ x \in [2,\ 3] & 12.4(x-2)^3 + 37.2(x-2)^2 + 40.2(x-2) + 16 \end{cases}$$

b) Tabulando:

x	P(x)	S(x)	Error=P(x)-S(x)|
0	0	0	0
0.25	0.0039	0.1562	0.1523
0.5	0.0625	0.35	0.2875
0.75	0.3164	0.6187	0.3023
1	1	1	0
1.25	2.4414	1.7125	0.7289
1.50	5.0625	3.7	1.3625
1.75	9.3789	8.0875	1.2914
2	16	16	0
2.25	25.6289	28.1812	2.5523
2.5	39.0625	43.85	**4.7875**
2.75	57.1914	61.8438	4.6523
3	81	81	0

Se observa que el máximo error es 4.7875.

c) El mayor error se registra en el tercer intervalo.

Problema 3

Un sistema dinámico presenta la siguiente respuesta en el tiempo:

t(seg)	0	0.2	0.4	0.6	0.8	1.2	1.6	2
y(m)	0.871	0.921	0.952	0.972	0.994	0.999	0.999	0.999

a) Realice un ajuste por mínimos cuadrados usando la función: $y = 1 - ae^{-bt}$. Indique a su criterio que tan buena es la función de ajuste obtenida?
b) Determine para que tiempo el sistema alcanza el 95 % de su posición máxima.

Solución

a) Agrupando convenientemente:

$$1 - y = ae^{-bt}$$
$$Ln(1-y) = -bt + Ln(a)$$
$$Y = BT + A$$

Realizando un ajuste lineal para la tabla:

T=t	0	0.2	0.4	0.6	0.8	1.2	1.6	2
Y=Ln(1-y)	-2.0479	-2.5383	-3.0366	-3.5756	-5.1160	-6.9078	-6.9078	-6.9078

Se obtiene:

B = -2.8174 A = -2.2349

Y = -2.8174 T - 2.2349

De donde se obtiene:

b=-B=2.8174

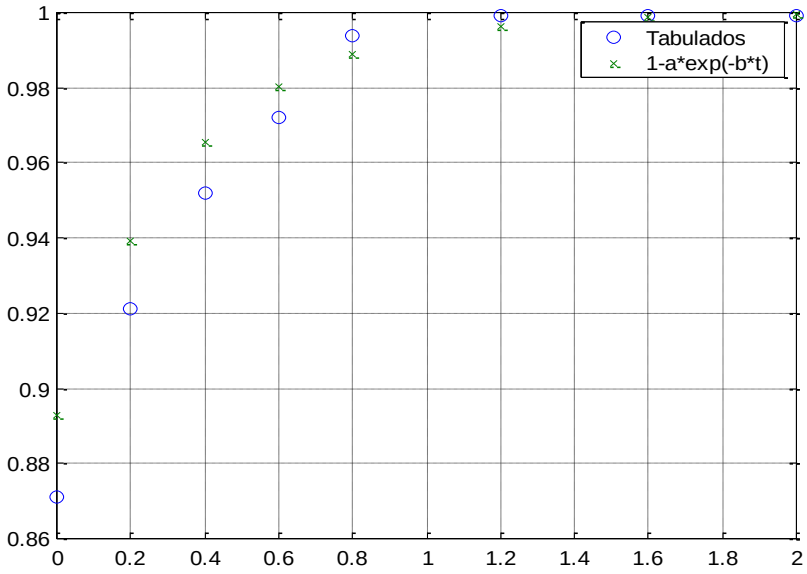
$a=e^{A}=0.1070$

Por lo tanto la función de Ajuste será: $y = 1 - 0.1070e^{-2.8174t}$

Tabulando:

t	0	0.2	0.4	0.6	0.8	1.2	1.6	2
y	0.871	0.921	0.952	0.972	0.994	0.999	0.999	0.999
$\hat{y} = 1 - 0.1070e^{-2.8174t}$	0.893	0.939	0.965	0.980	0.989	0.996	0.999	0.9996

Se observa que los resultados son satisfactorios, con error mas grande para los primeros puntos de la tabla.



b) Teniendo en cuenta que el y_{max} tiende a 1:

$y = 1 - 0.1070e^{-2.8174t}$

$0.95 = 1 - 0.1070e^{-2.8174t}$

$t \approx 0.27\ seg.$

5. Ejercicios Propuestos

1. Escriba una función que resuelva el spline natural a partir de la data (x,y), deberá obtener vectores que contengan a los coeficientes a_i, b_i, c_i, d_i.
   function [a,b,c,d]=myspline(x,y)

……..

……..

……..

……..

……..

2. Dada la siguiente tabla de diferencias divididas:

x	y	y[,]	y[,,]	y[,,,]
a	1			
		2		
1	3		h	
		f		c
2	5		2	
		g		
d	11			

El producto de c y d es: …….

3. En la interpolación de una función *f* que pasa por los puntos $x_i = 2i, \quad i = 0,1,2$. Se sabe que f(0)=-4 , f(4)=4 y f[2,4]=6. Hallar f[0,2,4]
   a) 6 b) -6 c) 2 d) -2 e) 0.

4. Sean los datos:

x	0	1	2
y	a	b	c

   Al usar el polinomio de Lagrange, ¿cuál será el polinomio $L_1(x)$ en comandos de MATLAB.
   a) L=poly([2 0])/polyval(poly([0 2],1))
   b) L=poly([2 0])/poly([0 2],1)
   c) L=polyval([0 2],1)/poly([0 1 2],1)
   d) L=poly ([0 2])/polyval(poly([0 2]),1)
   e) N.A.

5. Sean los siguientes puntos: (0,-1); (1,1); (2,9) y (3,29); por el cual se puede construir un polinomio interpolante de la forma:
   $p(x) = a + b(x-1) + c(x-1)(x-3) + d(x-1)(x-3)(x-2)$,
   entonces $a+b+c+d$ será:

   a) 7 b) 5 c) 22 d) 18 e) N.A.

6. Determinar si la siguiente función es un Spline cúbico en el intervalo [0,2]
$$S(x) = \begin{cases} (x-1)^3 & x \in [0,1] \\ 2(x-1)^3 & x \in [1,2] \end{cases}$$
   Justificar correctamente.

   ……
   ……
   ……

7. Determine los valores de a,b y c de modo que la siguiente función es un spline cúbico
$$S(x) = \begin{cases} \mathbf{x^3} & x \in \mathbf{[0,1]} \\ \mathbf{0.5(x-1)^3 + a(x-1)^2 + b(x-1) + c} & x \in \mathbf{[1,3]} \end{cases}$$

8. Aproximar la función de la tabla

x	0	1	2	3
y	1	1	1.7	2.5

   Por la función $f(x) \approx \dfrac{a + x^2}{b + x}$
   ¿Cuál es valor de la constante a?

	Curso	**Métodos Numérico**	**Código : MB536**
	Tema	**Diferenciación e Integración Numérica**	
	Practica	**07**	
	Profesores	**Garrido Juárez, Rosa** **Castro Salguero, Robert** **Pantoja Carhuavilca, Hermes** **Obregón Ramos, Máximo** **Ruiz Lizama, Edgar**	

1. Objetivos

Estudiar y aplicar los métodos para la aproximación numérica de derivadas e integrales de funciones. La función $f(x)$ puede conocerse en forma analítica o solo en forma discreta (tablas).

2. Fundamento Teórico

Fórmulas para la primera derivada:

Hacia delante

$$f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}$$

$$f'(x) = \frac{-3f(x)+4f(x+h)-f(x+2h)}{2h}+\mathcal{O}(h^2)$$

$$f'(x) = \frac{3f(x)-4f(x-h)+f(x-2h)}{2h}+\mathcal{O}(h^2)$$

$$f'(x) = \frac{f(x-2h)-8f(x-h)+8f(x+h)-f(x+2h)}{12h}+\mathcal{O}(h^4)$$

$$f'(x) = \frac{-25f(x)+48f(x+h)-36f(x+2h)+16f(x+3h)-3f(x+4h)}{12h}+\mathcal{O}(h^4)$$

Central

$$f'(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2h}$$

Fórmulas para la segunda derivada:

Hacia delante

$$f''(x_0) \approx \frac{f(x_0+2h)-2f(x_0+h)+f(x_0)}{h^2}$$

Central

$$f''(x_0) \approx \frac{f(x_0+h)-2f(x_0)+f(x_0-h)}{h^2}$$

La derivada puede aproximarse a su vez por un cociente en diferencias, pues se tiene que

$$f[x_0, x_1, \cdots, x_n, x_{n+1}] = \frac{f^{(n+1)}(\eta)}{(n+1)!}$$

para cierto η en el menor intervalo que contiene a $x_0, x_1, ..., x_{n+1}$.

Integración Numérica

Mostramos algunas fórmulas cerradas de Newton-Cotes más comunes para cualquier función *f(x).*

Formulas de Newton-Cotes Cerradas

Regla de los trapecios (n = 1)

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\left[f(a)+f(b)\right] \text{ o también}$$

$$\boxed{\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}\left(f(x_0)+f(x_1)\right)-\frac{h^3}{12}f''(\varepsilon) \quad donde\ x_0 < \varepsilon < x_1}$$

Regla de Simpson un tercio (n = 2)

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a)+4f(\frac{a+b}{2})+f(b)\right] \text{ o también:}$$

$$\boxed{\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}\left(f(x_0)+4f(x_1)+f(x_2)\right)-\frac{h^5}{90}f^{(4)}(\varepsilon) \quad donde\ x_0 < \varepsilon < x_2}$$

Regla de Simpson tres-octavos (n = 3)

$$\int_a^b f(x)dx \approx \frac{b-a}{8}\left[f(a)+3f(\frac{2a+b}{3})+3f(\frac{a+2b}{3})+f(b)\right]$$

o también:

$$\boxed{\int_{x_0}^{x_3} f(x)dx = \frac{3h}{8}\left(f(x_0)+3f(x_1)+3f(x_2)+f(x_3)\right)-\frac{3h^5}{80}f^{(4)}(\varepsilon) \quad donde\ x_0 < \varepsilon < x_3}$$

Regla de Milne (n = 4)

$$\int_a^b f(x)dx \approx \frac{b-a}{90}\left[7f(a)+32f(\frac{3a+b}{4})+12f(\frac{a+b}{2})+32f(\frac{a+3b}{4})+7f(b)\right]$$

$$\boxed{\int_{x_0}^{x_4} f(x)dx = \frac{2h}{45}\left(7f(x_0)+32f(x_1)+12f(x_2)+32f(x_3)+7f(x_4)\right)-\frac{8h^7}{945}f^{(6)}(\varepsilon) \quad donde\ x_0 < \varepsilon < x_4}$$

Formulas de Newton-Cotes abiertas:

- **Regla de Punto medio (n=0)**

$$\boxed{\int_{x_0}^{x_2} f(x)dx = 2h\left(f(x_1)\right)+\frac{h^3}{3}f''(\varepsilon) \quad donde\ x_0 < \varepsilon < x_2}$$

- **n=1**

$$\boxed{\int_{x_0}^{x_3} f(x)dx = \frac{3h}{2}\left(f(x_1)+f(x_2)\right)+\frac{3h^3}{4}f''(\varepsilon) \quad donde\ x_0 < \varepsilon < x_3}$$

- **n=2**

$$\boxed{\int_{x_0}^{x_4} f(x)dx = \frac{4h}{3}\left(2f(x_1)-f(x_2)+2f(x_3)\right)+\frac{14h^5}{45}f^{(4)}(\varepsilon) \quad donde\ x_0 < \varepsilon < x_4}$$

- **n=3**

$$\int_{x_0}^{x_5} f(x)dx = \frac{5h}{24}\left(11f(x_1) + f(x_2) + f(x_3) + 11f(x_4)\right) + \frac{95h^5}{144} f^{(4)}(\varepsilon)$$
$$donde\ x_0 < \varepsilon < x_5$$

Cuadratura de Gauss-Legendre:

Consideremos la cuadratura Gaussiana para evaluar:

$$I = \int_a^b f(t)dt$$

Donde [a,b]≠[-1,1], los límites de integración debe ser [-1,1] por lo cual recurrimos a un cambio de variable:

$$t = \frac{(b-a)x + (a+b)}{2} \qquad \text{y} \qquad dt = \frac{(b-a)}{2}dx$$

Reemplazando tendremos:

$$\int_a^b f(t)dt = \frac{(b-a)}{2}\int_{-1}^{1} f\left(\frac{(b-a)x + (b+a)}{2}\right)dx = \frac{(b-a)}{2}\sum_{i=1}^{n} c_i f(x_i) + E$$

Siendo x_i las raíces o ceros del polinomio de Legendre de grado "*n*", además:
A continuación se muestra una tabla conteniendo los factores de peso $\mathbf{c_i}$ y los ceros del polinomio de Legendre para $\mathbf{(x_i)}$ para diversos valores de **"N".**

Tabla de Gauss Legendre

N	x_i	c_i
1	0.0	2.0
2	-0.577350269189626 +0.577350269189626	1.0 1.0
3	-0.774596669241483 0.0 +0.774596669241483	0.555555555555556 0.888888888888889 0.555555555555556
4	-0.861136311594053 -0.339981043584856 +0.339981043584856 +0.861136311594053	0.347854845137454 0.652145154862546 0.652145154862546 0.347854845137454
5	-0.906179845938664 -0.538469310105683 0.0 +0.538469310105683 +0.906179845938664	0.236926885056189 0.478628670499366 0.568888888888889 0.478628670499366 0.236926885056189
6	-0.932469514203152 -0.661209386466265 -0.238619186083197 +0.238619186083197 +0.661209386466265 +0.932469514203152	0.171324492379170 0.360761573048139 0.467913934572691 0.467913934572691 0.360761573048139 0.171324492379170

Donde N es el número de puntos

3. Parte práctica

Ejemplo 1

La función "diff" permite el cálculo de derivadas exactas:

```
» syms x
» f=exp(x)*sin(x)
» fd=diff(f)
        fd =exp(x)*sin(x)+exp(x)*cos(x)
 » subs(fd,1)   % evaluando la derivada en x=1
        ans =3.7560
```

Ejemplo 2

Dados los siguientes puntos obtener la derivada a partir de un polinomio interpolante:

```
» x=[0 0.1 0.2 0.4 0.5 0.55]
» y=[1.12 1.28 1.55 1.88 1.77 1.66]
» p=polyfit(x,y,length(x)-1)
» dp=polyder(p)
» xi=[0 0.25 0.50]    % Evaluará p'(0)    p'(0.25)    p'(0.50)
» dpxi=polyval(dp,xi)
```

Ejemplo 3

Obtener la derivada a partir de puntos calculados con una función:

```
%pruebadiff1.m
f=inline('x.^2.*exp(x)')
df=inline('(x.^2+2*x).*exp(x)')
x=0:0.25:1
y=f(x)
p=polyfit(x,y,length(x)-1)
dp=polyder(p)
xi=[0 0.25 0.50 0.8 1 1.1 1.2]
dpxi=polyval(dp,xi)
dfxi=df(xi)
err=abs(dpxi-dfxi)
```

Ejemplo 4

Uso de formulas de diferenciación de dos puntos

%pruebadiff2.m

```
f=inline('x.^2.*exp(x)')
df=inline('(x.^2+2*x).*exp(x)')
x=1
h=0.1
dfa=(f(x+h)-f(x))/h
dfe=df(x)
err=abs(dfe-dfa)

Probar para diferentes valores de h=0.1, 0.01, 0.001 hasta encontrar
el h critico donde el error empieza ha aumentar.
```

Ejercicio.- Aproxime con la derivada central **f'(x)=(f(x+2h)-f(x-h))/(2h),** y determine el valor del h crítico para este caso.

Ejemplo 5
Cálculo de Integrales indefinidas

Se utiliza el comando **int**.

Ejemplo:

» syms t % variable simbólica
» f=inline('cos(x)*exp(x)');
» int(f(t),t)
» pretty(ans) % reescribimos la solucion de forma 'elegante'

Ejemplo 6
Integrales Definidas

La misma instrucción, **int**, permite realizar la integración definida. A modo de Ejemplo:

```
» int((t^2+4*t)/(t^4-2*t^2+1),t,2,5)
   ans =13/16+1/4*log(2)
» double(ans)
  ans =  0.9858
```

Ejemplo 7
La función "quad" permite obtener integrales en forma numérica:

```
» f=inline('exp(x).*sin(x)')
» I=quad(f,1,2)
I =    4.4875
```

Ejemplo 8

Aproxime : $\int_0^1 e^x\left(x^2+2x\right)dx$, usando la regla del trapecio (h=1)

```
» f1=inline('exp(x).*(x.^2+2*x)')
» h=1
» Ie=exp(1)
» I=h/2*(f1(0)+f1(1))
       I =    4.0774
» err=abs(Ie-I)
       err = 1.3591
```

Usando la regla del trapecio con h=1/2

```
» h=0.5
» I=h/2*(f1(0)+2*f1(0.5)+f1(1))
       I =   3.0692
» err=abs(Ie-I)
      err =   0.3509
```

Ejercicio.- aplique el trapecio compuesto con h=1/8

Ejemplo 9
Escriba una función para la fórmula del trapecio compuesta:

```
% trapecio.m
function I=trapecio(fname,a,b,n)
h=(b-a)/n;
x=a:h:b;
f=feval(fname,x);
I=h/2*(f(1)+2*sum(f(2:n))+f(n+1));
```

» I=trapecio(f1,0,1,8)
I = 2.74043839175033

» err=abs(I-Ie)

err = 0.02215656329129

completar la siguiente tabla:

n	I	error
1 2 4 8 16 32 64	2.74043839175033	0.02215656329129

Ejemplo 10
Resuelva la integral anterior usando la regla de Simpson 1/3, h=0.5
» h=0.5
» I=h/3*(f1(0)+4*f1(0.5)+f1(1))
I = 2.73307530647963
Ejercicio.- Evalúe la integral con h=1/8.

Ejemplo 11
Escriba una función para la formula compuesta de Simpson 1/3:

```
% sim13.m
% Formula compuesta de Simpson 1/3
% n : Numero total de particiones debe ser par
function I=sim13(fname,a,b,n)
if rem(n,2)==0
   h=(b-a)/n;
   x=a:h:b;
   f=feval(fname,x);
   I=h/3*(f(1)+4*sum(f(2:2:n))+ 2*sum(f(3:2:n-1))+f(n+1));
else
   disp('Lo siento n debe ser par!!!')
end
```

»I2=sim13(f1,0,1,2)
I2 = 2.7331
Complete la tabla anterior

Ejemplo 12
Aproxime usando la regla del rectángulo:

$$\int_0^1 \frac{1}{\sqrt{-\log(x)}} dx$$

» f2=inline('1./sqrt(-log10(x))')
» h=0.5
» I=2*h*f2(0.5)

I = 1.82261572880500

» h=0.25
» I=2*h*f2(0.25)+2*h*f2(0.75)

I = 2.05895221855840

Ejercicio.- Evalúe la integral con h=1/16.

Ejemplo 13

Elabore la siguiente función:
% rectangulo.m
```
function I=rectangulo(fname,a,b,n)
```

Luego calcule la integral anterior con n=2, 4, 8, 16, 32, 64, 128, 256.

Ejemplo 14

$$\int_{-1}^{1}\left(x^2+2x\right)e^x dx$$

Aproximar, mediante cuadratura de Gauss, con n=1, 2, 3, 4:

» f1=inline('exp(x).*(x.^2+2*x)')
» I1=2*f1(0)
I1 = 0
» I2=f1(-0.5774)+f1(0.5774)
I2 = 2.1900
» Ie=exp(1)-exp(-1)
Ie = 2.3504
» err=abs(I2-Ie)
err = 0.1604

Ejemplo 15

Utilizando la cuadratura de Gauss-Legendre (n=1, 2, 3, 4), estime la siguiente integral:

$$I=\int_1^{1.5} e^{-t^2}dt$$

Solución:

a=1, *b*=2, haciendo $t=\dfrac{(1.5-1)x+(1.5+1)}{2}=\dfrac{0.5x+2.5}{2}=\dfrac{x+5}{4}$

$$dt=\frac{dx}{4}$$

$$\int_1^{1.5} e^{-t^2} dt = \int_{-1}^{1} e^{-\left(\frac{x+5}{4}\right)^2} \left(1/4\right) dx$$

$$F(x) = e^{-\left(\frac{x+5}{4}\right)^2} \left(\frac{1}{4}\right)$$

```
» f2=inline('0.25*exp(-((x+5)/4)^2)')
» I1=2*f2(0)
        I1 =  0.10480569357555
» I2=f2(-0.5774)+f2(0.5774)
        I2 =   0.10940104496734
» es=abs(I2-I1)
    es =  0.00459535139179
```

Ejemplo 16

Los Polinomios de Legendre se pueden obtener recursivamente con la siguiente

$$P_0(x) = 1$$

formula: $P_1(x) = x$

$$P_{n+1} = \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x)$$

Obteniéndose la siguiente tabla:

n	**$P_n(x)$**
0	1
1	x
2	$(3x^2-1)/2$
3	$(5x^3-3x)/2$
4	$(35x^4-30x^2+3)/8$
5	$(63x^5-70x^3+15x)/8$

Escriba una función para generar el polinomio de Legendre para cualquier valor "n":

```
% legendre.m
function p=legendre(n)
if n==0
   p=1;
elseif n==1
   p=[1 0];
else
   p=((2*n-1)*[legendre(n-1) 0]-(n-1)*[0 0 legendre(n-2)])/n;
end
```

```
» p=legendre(2)
p =    1.5000         0   -0.5000   % 1.5 x^2 – 0.5
```

Ejemplo 17

Escriba una función que genera las tablas de la cuadratura de Gauss-Legendre a partir de cualquier entero "n", donde los valores "x_i" son las raíces del polinomio de Legendre y los pesos "c_i"se obtienen con la siguiente relación:

$$c_i = \frac{2}{(P'_n(x_i))^2(1-x_i^2)}$$

```
% gauss_legendre.m
function [x,c]=gauss_legendre(n)
p=legendre(n);
x=roots(p);
c=2./(polyval(polyder(p),x)).^2./(1-x.^2);
```

```
» [x,c]=gauss_legendre(3)
x =
         0
    0.7746
   -0.7746
c =
    0.8889
    0.5556
    0.5556
```

Ejemplo 18

Escriba una función para obtener la integral mediante la cuadratura de Gauss-Legendre, con la siguiente cabecera:

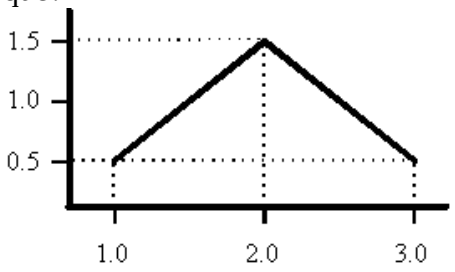
function I= int_gauss(f, a, b, n)

```
%int_gauss.m
function I=int_gauss(f,a,b,n)
[x,c]=gauss_legendre(n);
xx=feval(f,(b-a)/2*x+(b+a)/2);
I=(b-a)/2*sum(c.*xx);
```

```
» f1=inline('exp(x).*(x.^2+2*x)')
» I=int_gauss(f1,0,1,2)
I =    2.7085
```

4. Ejercicios Propuestos

1. Calcule numéricamente la integral de la figura adjunta empleando la regla del trapecio y la regla de Simpson. ¿Qué método es más exacto en este caso? ¿Por qué?



2. Aproximar la siguiente integral $I = \int_0^1 \frac{1-x^2}{1+x^2}dx$ utilizando la siguiente formula:

$$\int_{-1}^{1} f(x)dx \approx \frac{1}{9}\left[5f(-\sqrt{3/5}) + 8f(0) + 5f(\sqrt{3/5})\right]$$

……

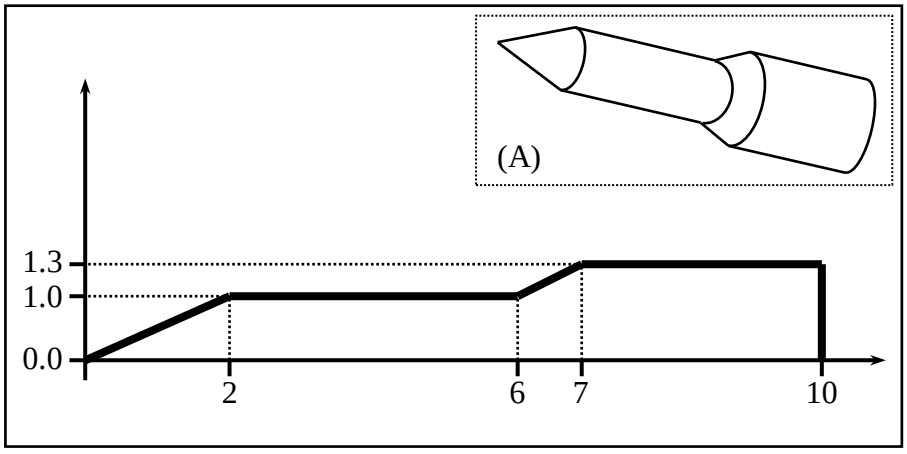
……

3. ¿Cuantos intervalos necesitamos para aproximar la integral $\int_1^3 \ln(x)dx$ por el método de Simpson con un error $\varepsilon < 0.001$

……
……

4. La rotación alrededor del eje *x* del perfil representado en la gráfica adjunta genera un sólido de revolución similar al mostrado en la figura (A). El volumen del sólido de revolución engendrado por rotación alrededor del eje *x* del perfil representado en la gráfica viene dado por la expresión:

$$V = \pi \int_a^b [f(x)]^2 dx$$



Calcule el volumen de dicho sólido empleando la regla del trapecio.

……
……

5. Hallar $\int_0^1 e^{x^2} \sin x dx$, usando el método Romberg usando los pasos de 1 y 0.5

……
……

6. Encuentra a, b y c de tal manera que la siguiente fórmula de integración sea exacta para cualquier polinomio de grado 2 o menor.

$$\int_{-2h}^{2h} f(x)\,dx \approx a\,f(-h) + b\,f(0) + c\,f(h)$$

……
……

7. Dada la integral $\int_1^4 \frac{1}{x} dx$, determine el número mínimo de sub-intervalos necesarios para que se obtenga el valor de la integral por el método del trapecio con un error inferior a 0.01

……
……

	Curso	**Métodos Numéricos**	**Código : MB536**
	Tema	**Ecuaciones Diferenciales Ordinarias**	
	Practica	**08**	
	Profesores	**Garrido Juárez, Rosa** **Castro Salguero, Robert** **Pantoja Carhuavilca, Hermes** **Obregón Ramos, Máximo** **Ruiz Lizama, Edgar**	

1. Objetivos

Estudiar y aplicar los diversos métodos iterativos, para la solución de Ecuaciones Diferenciales Ordinarias con problema de valor inicial y problema de valor frontera.

2. Fundamento Teórico

Problemas de Valor Inicial

Por lo general, la solución exacta de un problema de valor inicial para EDO es imposible ó difícil de obtener en forma analítica. Normalmente no queda otro remedio que la búsqueda de una solución numérica

Vamos a presentar métodos diferentes para realizar esto, empezando con el método más sencillo. Pretendemos resolver:

$$\begin{cases} y'(t) = f(t, y(t)) \quad , \quad t_0 = a < t < b \\ y(t_0) = \alpha \quad (Condición\ Inicial) \end{cases}$$

Tomamos el tamaño de paso h>0 (h = (b - a)/N)

definiendo $t_i = t_0 + ih \quad , i = 0,1,2,\ldots,N-1$

2.1.1 Método de Euler

Progresivo:

$$\boxed{y_{i+1} = y_i + hf(t_i, y_i), \qquad i = 0,1,\ldots N-1}$$

Regresivo:

$$\boxed{y_{i+1} = u_i + hf(t_{i+1}, y_{i+1}), \qquad i = 0,1,\ldots N-1}$$

2.1.2 Método de Taylor de orden 2

$$\boxed{y_{j+1} = y_j + hf(t_j, y_j) + \frac{h^2}{2} f'(t_j, y_j)}$$

2.1.3 Método de Runge-Kutta

De orden 2

$$k_1 = hf(t_j, y_j)$$

$$k_2 = hf(t_{j+1}, y_j + k_1)$$

$$y_{j+1} = y_j + \frac{1}{2}[k_1 + k_2]$$

De orden 4

$$k_1 = hf(t_j, y_j)$$

$$k_2 = hf(t_j + \frac{h}{2}, y_j + \frac{k_1}{2})$$

$$k_3 = hf(t_j + \frac{h}{2}, y_j + \frac{k_2}{2})$$

$$k_4 = hf(t_j + h, y_j + k_3)$$

$$y_{j+1} = y_j + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

2.2 Problema del Valor Frontera

2.2.1 Método del Disparo

Sea el problema de valor frontera en una EDO de segundo orden

$$u'' = g(t, u, u')$$

$$u(t_0) = u_0$$

$$u(b) = B$$

Se puede resolver como un problema de valor inicial donde la pendiente *s* va variando hasta que ***u(b)*** sea muy cercano a ***B***

$$u'' = g(t, u, u')$$

$$u(t_0) = u_0$$

$$u'(t_0) \approx s$$

Ejemplo

Resolver:

$$y'' - y' - 2y = 0$$

$$y(0) = 0.1$$

$$y(0.5) = \mathbf{0.283}$$

$$t \in [0, 5]$$

Solución:

Se resuelve el siguiente problema con las condiciones iniciales:

$$y'' - y' - 2y = 0$$

$$y(0) = 0.1$$

$$y'(0) = s$$

Elijamos un valor inicial de $s = s_0 = \frac{\Delta y}{\Delta t} = \frac{0.283 - 0.1}{0.5 - 0} = 0.3660$

Sea h=0.1

Aplicando RK4 al PVI,

$$y'' - y' - 2y = 0$$

$$y(0) = 0.1$$

$$y'(0) = 0.3660$$

obtenemos:

y =

0	0.1000	0.3660
0.1000	0.1397	0.4295
0.2000	0.1864	0.5088
0.3000	0.2420	0.6071
0.4000	0.3086	0.7285
0.5000	**0.3887**	0.8780

$y_N = y_N(s_o)$

Se obtiene $y_N = y_N(s_o)$ y comparamos con $y(0.5) = 0.283$.

Se elige un segundo valor para $s = s_1$

$$s_1 = s_0 + \frac{B - y_N}{b - t_0} = 0.3660 + \frac{0.283 - 0.3887}{0.5 - 0} = 0.1547$$

Aplicando RK4 al PVI,

$y'' - y' - 2y = 0$

$y(0) = 0.1$

$y'(0) = 0.1547$

y =

0	0.1000	0.1547
0.1000	0.1174	0.1937
0.2000	0.1390	0.2409
0.3000	0.1659	0.2981
0.4000	0.1990	0.3677
0.5000	0.2399	0.4523

$y_N = y_N(s_1)$

Se obtiene $u_N = u_N(s_1)$ y comparamos con $y(0.5) = 0.1547$.

Utilice interpolación lineal a fin de obtener elecciones subsecuentes valores para *s*, esto es:

$$s_{k+2} = s_k + (s_{k+1} - s_k)\frac{0.283 - y_N(s_k)}{y_N(s_{k+1}) - y_N(s_k)} \quad k = 0,1,2,\ldots$$

Con cada S_k resolvemos el problema de valor inicial y comparamos $u_N(s_k)$ con 0.283

Hallando S_3

$$s_2 = s_0 + (s_1 - s_0)\frac{0.283 - y_N(s_0)}{y_N(s_1) - y_N(s_0)} = 0.3660 + (0.1547 - 0.3660) * \frac{0.283 - 0.3887}{0.2399 - 0.3887} =$$

$$= 0.2159$$

Aplicando RK4 al PVI,

$y'' - y' - 2y = 0$

$y(0) = 0.1$

$y'(0) = 0.2159$

y =

0	0.1000	0.2159
0.1000	0.1238	0.2620
0.2000	0.1527	0.3185
0.3000	0.1879	0.3876
0.4000	0.2308	0.4722
0.5000	0.2830	0.5756

$y_N = y_N(s_3)$

Se detiene cuando $|y_N(s_k) - B|$ sea suficientemente pequeño (Criterio de convergencia). En este caso $|y_N(s_3) - 0.2830| \approx 0$

2.2.2 Método de las diferencias finitas

Dado el problema de valor inicial de segundo orden con valor frontera

$$u'' + p(t)u' + q(t)u = r(t) \qquad a \le t \le b$$
$$u(a) = \alpha \qquad u(b) = \beta$$

se requiere aproximar u', u'' usando diferencias centrales.

Para lo cual seleccionamos un número N>0 y dividimos el intervalo [a,b] en (N+1) sub-intervalos iguales., cuyos puntos son los puntos de la retícula

$t_i = a + ih$, para i=0,1,..N+1, donde $\boldsymbol{h = \frac{(b-a)}{N+1}}$.

Esto nos lleva a formar un sistema lineal de orden NxN, que puede ser resuelto por el método para resolver SEL discutidos anteriormente.

En los puntos interiores de la retícula t, i= 1,2,.. ,N, la ecuación diferencial a aproximar es:

$$u''(t_i) = p(t_i)u'(t_i) + q(t_i)u(t_i) + r(t_i) \qquad (1)$$

Sabemos que:

$$u''(t_i) \approx \frac{u(t_{i+1}) - 2u(t_i) + u(t_{i-1})}{h^2}$$

$$u'(t_i) \approx \frac{u(t_{i+1}) - u(t_{i-1})}{2h}$$

Resultando:

$$\frac{u(t_{i+1}) - 2u(t_i) + u(t_{i-1})}{h^2} = p(t_i)\frac{u(t_{i+1}) - u(t_{i-1})}{2h} + q(t_i)u(t_i) + r(t_i) \qquad (2)$$

en la frontera:

$u(a) = \alpha \qquad u(b) = \beta$

Definiendo:

$w_0 = \alpha \qquad w_{N+1} = \beta$

la ec(2) queda:

$$\frac{w_{i+1} - 2w_i + w_{i-1}}{h^2} = p(t_i)\frac{w_{i+1} - w_{i-1}}{2h} + q(t_i)w_i + r(t_i) \qquad (3)$$

La ec (3) se puede escribir como:

$$\left(1+\frac{h}{2}p(t_i)\right)w_{i-1}+\left(-h^2q(t_i)-2\right)w_i+\left(1-\frac{h}{2}p(t_i)\right)w_{i+1}=h^2r(t_i)$$

Haciendo:

$$A_i=\left(1+\frac{h}{2}p(t_i)\right)$$

$$B_i=\left(-h^2q(t_i)-2\right)$$

$$C_i=\left(1-\frac{h}{2}p(t_i)\right)$$

$$D_i=h^2r(t_i)$$

$$\begin{bmatrix} B_1 & C_1 & & & & \\ A_2 & B_2 & C_2 & & & \\ & & & & & \\ & & A_{N-1} & B_{N-1} & C_{N-1} & \\ & & & A_N & B_N & \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \\ \\ w_{N-1} \\ w_N \end{bmatrix}=\begin{bmatrix} D_1-A_1w_o \\ D_2 \\ \\ D_{N-1} \\ D_N-C_Nw_{N+1} \end{bmatrix}$$

La solución del sistema lineal, se puede realizar por reducción de Crout.

Ejemplo:
Resolver aplicando diferencias finitas:

y'' = (-2/x)y'+(2/x^2)y+sin(Lnx)/x^2
y(1)=1
y(2)=2
h=0.2

Solución

y'' = p(x)y'+q(x)y+r(x)

$$p(x_i)=-\frac{2}{x_i}$$

$$q(x_i)=\frac{2}{x_i^{\,2}}$$

$$r(x_i)=\frac{\sin(\ln(x_i))}{x_i^{\,2}}$$

$$A_i=\left(1+\frac{h}{2}p(x_i)\right)$$

$$B_i=-\left(h^2q(x_i)+2\right)$$

$$C_i=\left(1-\frac{h}{2}p(x_i)\right)$$

$$D_i=h^2r(x_i)$$

$$\begin{bmatrix} B_1 & A_1 & & & & \\ C_2 & B_2 & A_2 & & & \\ & & & & & \\ & & C_{N-1} & B_{N-1} & A_{N-1} & \\ & & & C_N & B_N & \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \\ w_{N-1} \\ w_N \end{bmatrix} = \begin{bmatrix} D_1 - C_1 w_0 \\ D_2 \\ \\ D_{N-1} \\ D_N - A_N w_N \end{bmatrix}$$

la solución con diferencias finitas

t(k)	y(k)=X1
1.00000000000000	1.00000000000000
1.20000000000000	1.18692106203224
1.40000000000000	1.38127313638853
1.60000000000000	1.58226407575024
1.80000000000000	1.78883227403986
2.00000000000000	2.00000000000000

3. Instrucciones básicas en MATLAB

Solución de Ecuaciones Diferenciales	
Ode23	**Método de Runge-Kutta de orden 2/3**
Ode45	**Método de Runge-Kutta-Fehlberg de orden 4/5**

Ejemplo:

```
function dydt = Ej1(t,y)
dydt=exp(t)/((1+exp(t))*y(1));
```

En la ventana de comandos

```
>> [t,y]=ode45(@Ej1,[0 5],3);
>> plot(t,y)
```

dsolve('ec1',…,'ecn')	Resuelve el sistema de ecuaciones y condiciones Iniciales {ec1, . . ., ec2}

La letra D se utiliza para representar la derivación con respecto a la variable independiente, es decir, u' se escribe Du; las derivadas orden superior u'', u''', . . . se escriben D2u, D3u, . . .

Ejemplo:

Para resolver el problema de valores iniciales

$u' = 0.5 * u, \qquad u(0) = 0.25$

Solución

```
>> u = dsolve('Du = u/2','u(0) = 1/4')
u =
      1/4*exp(1/2*t)
```

4. Parte práctica

Considere la ecuación diferencial $y' = \frac{dy}{dx} = y(x^2 - 1)$ con y(0) = 1 y $x \in [0, 1]$

a) Calcule las soluciones aproximadas usando los métodos de Euler progresivo y regresivo con paso $h = 0.1$ Determine un máximo error de truncación.

b) Calcule la solución aproximada usando el método de Taylor de segundo orden con paso $h = 0.2$.

Solución

a) Método de Euler progresivo

$$y_{i+1} = y_i + hy'_i$$

$$y_{i+1} = y_i + hy_i(x_i^2 - 1)$$

Programa en MATLAB

```
function [z]=eulerp(f,a,b,y,h)
x=a:h:b;
n=length(x);
z=[x(1) y];
for i=1:n-1
   y=y+h*feval(f,x(i),y);
   z=[z ;x(i+1) y];
end
```

Probar:

```
>>f=inline('y.*(x.*x-1)','x','y ')
>>z=eulerp(f,0,1,1,0.1)   % tabla
>>x=z(:,1);y=z(:,2);
>>plot(x,y);
```

ii.- Método de Euler regresivo

$$y_{i+1} = y_i + hf(x_{i+1}, y_{+1i})$$

Sustituyendo para el presente caso tenemos:

$$y_{i+1} = y_i + hy_{i+1}((x_i + h)^2 - 1)$$

Como en esta ecuación podemos despejar y_{i+1} (¡)

$$y_{i+1} = \frac{y_i}{1 + h(1 - (x_i + h)^2)}$$

Tabla de Euler Progresivo

x_i	y_i
0.0	1.0000
0.1	0.9000
0.2	0.8109
0.3	0.7331
0.4	0.6663
0.5	0.6104
0.6	0.5646
0.7	0.5285
0.8	0.5015
0.9	0.4835
1.0	0.4743

Tabla de Euler Regresivo

x_i	y_i
0.0	1.0000
0.1	0.9099
0.2	0.8302
0.3	0.7610
0.4	0.7050
0.5	0.6530
0.6	0.6137
0.7	0.5840
0.8	0.5637
0.9	0.5532
1.0	0.5412

Cálculo del error máx. de truncación en los métodos de Euler (progresivo y regresivo)

$$\begin{aligned}\|T_h\| &\le \frac{h}{2}\times \sup_{x\in[0,1]}\left|\frac{d}{dx}\left[y(x).(x^2-1)\right]\right| \\ &= 0.05\times \sup_{x\in[0,1]}\left|y(x).\left[2x+(x^2-1)^2)\right]\right| \\ &\le 0.05\times y(0)\times 1 \\ &= 0.1\end{aligned}$$

b) Método de Taylor de segundo orden el valor de y_{i+1} es determinado por la expresión:

$$y_{i+1} = y_i + h.f(x_i, y_i) + \frac{h^2}{2} f'(x_i, y_i)$$

Haciendo la sustitución para este ejemplo tenemos:

$$y_{i+1} = y_i + h.y_i.(x_i^{\,2}-1) + \frac{h^2}{2} y_i.\left[2x_i + \left(x_i^{\,2}-1\right)^2\right]$$

Aplicando sucesivamente esta formula se obtiene la siguiente tabla de valores:

x_i	y_i
0.0	1.0000
0.2	0.8200
0.4	0.6842
0.6	0.5899
0.8	0.5344
1.0	0.5134

```
function [z]=taylor2(f,df,a,b,y,h)
x=a:h:b;
n=length(x);
z=[x(1) y];
for i=1:n-1
y=y+h*feval(f,x(i),y)+(h^2)/2*feval(df,x(i),y);
z=[z ;x(i+1) y];
end
```

Para probar taylor2.m

```
>>f=inline('y.*(x.*x-1)','x','y ')
>>df=inline('y*(2*x+(x*x-1)^2)')
>> z=taylor2(f,df,0,1,1,0.2)  % tabla
```

2. Resolver el siguiente problema diferencial con condiciones iniciales:

$$y'(t) = \frac{2y}{t} + t^2 e^t$$
$$y(1) = 0 \qquad , t \in [1,2]$$

Utilizar el método de Euler modificado usando un paso h = 0.25. Comparar con el valor exacto y(2) = 18.6831 y evaluar el error porcentual.

Solución:

Paso h = 0.25 $\Rightarrow$ y(2) = y_4

$y_e = y_i + hf(t_i, y_i)$

$t_{i+1} = t_i + h$

$y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, y_e)]$

$y_e = 0.67957 \quad t_1 = 1.25 \quad y_1 = 1.1574$

$y_e = 2.9838 \quad t_2 = 1.5 \quad y_2 = 3.8284$

$y_e = 7.6254 \quad t_3 = 1.75 \quad y_3 = 9.0192$

$y_e = 16.002 \quad t_4 = 2.0 \quad y_4 = 18.205$

ε = 18.6831 - 18.205 =0.4781 %εr = 2.6 %

3. Considérese el problema diferencial de condiciones iniciales :

$y' = 4e^{0.8x} - 0.5y$

y(0) = 2 $\quad x \in [0,3]$

Resolver por el algoritmo de Runge - Kutta de cuarto orden , tamaño de paso h = 1.5. Comparar la solución obtenida con la solución exacta y(3) = 33.6772 y evaluar el error

Solución:

$y' = f(x,y) = 4e^{0.8x} - 0.5y$

y(0) = 2 $\quad x \in [0,3]$

h = 1.5 $\Rightarrow$ y(3) = y_2

i = 0 $k_1 = 3, k_2 = 5.1635, k_3 = 4.3522, k_4 = 9.0163$

$y_1 = 2 + \frac{1.5}{6}(k1 + 2k2 + 2k3 + k4) = 9.7619$

i = 1 $k_1 = 8.3995 \quad k_2 = 16.168 \quad k_3 = 13.255 \quad k_4 = 29.271$

:

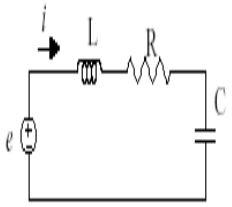
$y_2 = y1 + \frac{1.5}{6}[k1 + 2k2 + 2k3 + k4] = 33.891$

$\%\mathcal{E} = \frac{|33.891 - 33.6772|}{33.6772} * 100 = 0.634\%$

```
function [y]=rk4s(f,a,b,u,h)
t=a:h:b;
n=length(t);  y=[t(1) u];
for i=1:n-1
   k1=feval(f,t(i),u);
   k2=feval(f,t(i)+h/2,u+h*k1/2);
   k3=feval(f,t(i)+h/2,u+h*k2/2);
   k4=feval(f,t(i)+h,u+h*k3);
   u=u+h/6*(k1+2*k2+2*k3+k4);
   y=[y ;t(i+1) u];
end
```

```
>> f=inline('4*exp(0.8*x)-0.5*y','x','y ')
>> [y]=rk4s(f,0,3,2,1.5)
```

4. Considere el siguiente circuito eléctrico:



La ecuación diferencial para este circuito eléctrico es el siguiente:

$$L\frac{di}{dt} + Ri + \frac{1}{C}\int idt = e(t)$$

Dado que la carga eléctrica está definida como $q = \int idt$ la ecuación se puede escribir:

$$L\frac{d^2q}{dt^2} + R\frac{dq}{dt} + \frac{1}{C}q = e(t)$$

Determine el valor de la carga q en t$\in$ [0 1] con h =0.1, para el caso R =1,L = 0.5, C=1, e(t)=0.5

$q'' + 2q' + 2q = 1$, con $q(0)=0$, $q'(0) = 0$

Solución

En una ecuación diferencial de segundo orden el primer paso es su transformación en un sistema de dos ecuaciones de 1er. Orden. Por tal razón hacemos $u_1=q$ y

$$\begin{cases} u_1' = u_2 \\ u_2' = 1 - 2u_1 - 2u_2 \\ u_1(0) = 0 \\ u_2(0) = 0 \end{cases}$$

En **MATLAB**:

```
% fu.m
function [u_dot]=fu(t,u)
 u1=u(1); u2=u(2);
 f1=u2;
 f2=1-2*u1 -2*u2;
u_dot=[f1 f2];
```

a) Solución Mediante Euler para sistemas:

» [y]=eulerp('fu',0,1,[0 0],0.1)

y =

0	0	0
0.1000	0	0.1000
0.2000	0.0100	0.1800
0.3000	0.0280	0.2420
0.4000	0.0522	0.2880
0.5000	0.0810	0.3200
0.6000	0.1130	0.3398
0.7000	0.1470	0.3492
0.8000	0.1819	0.3500
0.9000	0.2169	0.3436
1.0000	0.2513	0.3315

b) Solución mediante Runge-Kutta 4 para sistemas:

» [y]=rk4s('fu',0,1,[0 0],0.1)

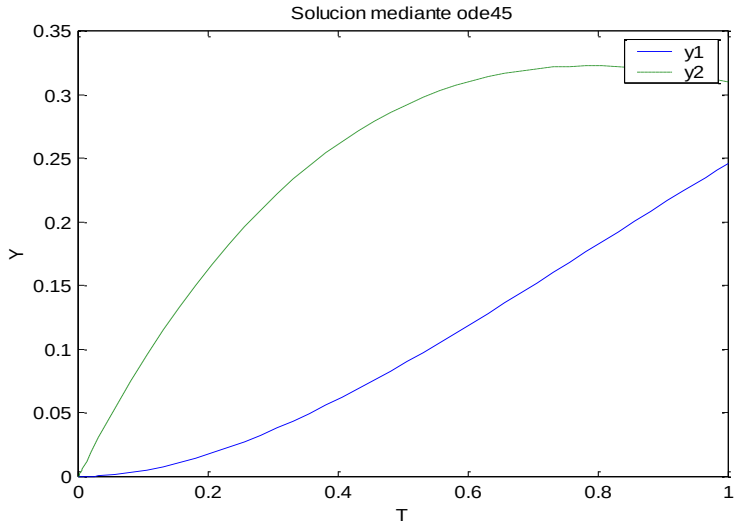
y =

0	0	0
0.1000	0.0047	0.0903
0.2000	0.0175	0.1627
0.3000	0.0367	0.2189
0.4000	0.0608	0.2610
0.5000	0.0885	0.2908
0.6000	0.1186	0.3099
0.7000	0.1501	0.3199
0.8000	0.1823	0.3223
0.9000	0.2144	0.3185
1.0000	0.2458	0.3096

c) Mediante la función "ode45" del MATLAB:

```
% fu1.m
function [u_dot]=fu1(t,u)
    u_dot=[u(2); 1-2*u(1)-2*u(2)];
```

```
% prueba ode.m

% [T, Y]=ode45('fu1',[To Tf],[y1o y2o])
[T, Y]=ode45('fu1',[0 1],[0 0])
plot(T,Y(:,1),'-',T,Y(:,2),'--')
title('Solucion mediante ode45')
xlabel('T')
ylabel('Y')
legend('y1','y2')
```



d) Solución mediante matemáticas simbólicas:

- Ecuación simple

```
» y=dsolve('Dy=2*t*y')

y =

C1*exp(t^2)

» y=dsolve('Dy=2*t*y','y(1)=1')

y =

1/exp(1)*exp(t^2)
```

- Ecuación de orden superior

```
» y=dsolve('D2y+2*Dy+2*y=1','y(0)=0','Dy(0)=0','x')

y =

1/2-1/2*exp(-x)*sin(x)-1/2*exp(-x)*cos(x)

» xx=0:0.1:1;
» yy=subs(y,xx)      % Substituye valores
```

» plot(xx,yy)

- Para un sistema

```
% test.m
[x,y]=dsolve('Dx=y,Dy=1-2*x-2*y','x(0)=0','y(0)=0')
% x =
% 1/2+exp(-t)*(-1/2*cos(t)-1/2*sin(t))
% y =
% exp(-t)*sin(t)
tt=0:0.1:1
xx=subs(x,tt)
% xx = 0 0.0047 0.0175 0.0367 0.0608 0.0885 0.1186 0.1501 0.1823 0.2144 0.2458
yy=subs(y,tt)
% yy = 0 0.0903 0.1627 0.2189 0.2610 0.2908 0.3099 0.3199 0.3223 0.3185 0.3096
plot(tt,xx,tt,yy)
```

5. (Tomado de Eduardo Raffo Lecca "Métodos Numéricos para Ciencias e Ingeniería con MATLAB", 2009). Considere a continuación el problema del péndulo simple, sujeto por una cuerda de longitud *L*, que aparece en la figura.

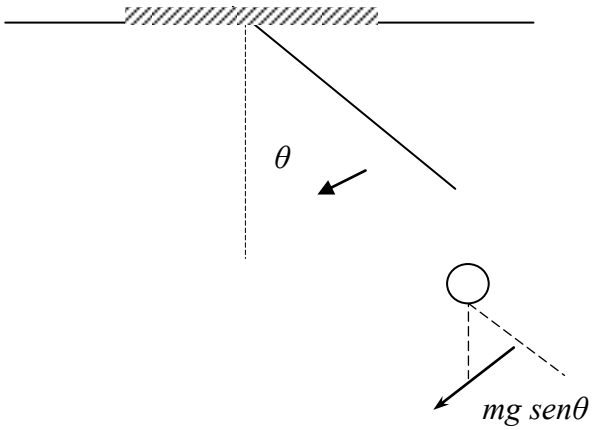


Figura: Péndulo simple

$$m\frac{d^2x}{dt^2} + mgsen\theta = 0$$

$$dx = Ld\theta$$

$$\frac{dx^2}{dt^2} = L\frac{d}{dt}\left(\frac{d\theta}{dt}\right) = L\frac{dw}{dt} = L\left(\frac{dw}{d\theta}.\frac{d\theta}{dt}\right)$$

$$= Lw\frac{dw}{d\theta}$$

La EDO es:

$$\frac{wdw}{d\theta} = -\left(\frac{g}{L}\right) sen\,\theta$$

$$\frac{dw}{d\phi} = \frac{-g}{L}\frac{\sin\phi}{w}, \qquad w(79^o) = 0.66488546$$

Donde:

w = velocidad angular en rad/sec,
ϕ = ángulo,
g = 32.2014 pies/sec^2,
L = 2.5 pies, $x \in [79^\circ \; a \; 65^\circ]$, y (79º) = 0.66488546

El archivo dwdfi.m es el siguiente:

```
function deriv = dwdfi(fi,w)
% motion of a simple pendulum
% E. Raffo Lecca
       global L;
       g=32.2014;
       %wLdw/dfi=-gsin(fi)
      deriv=(-g/L)*sin(fi)/w;
```

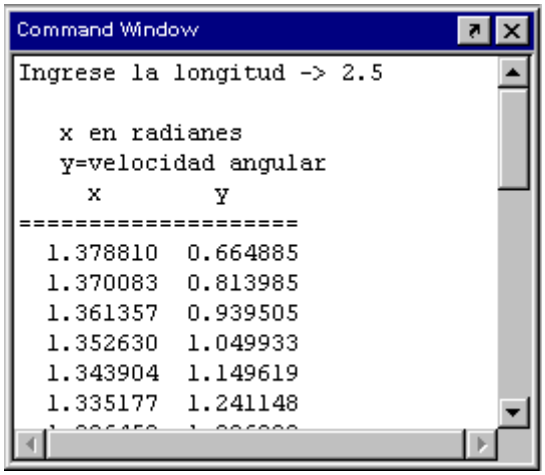
El programa *pedulum.m*, se presenta a continuación; en él se invoca a *RK4.m*.

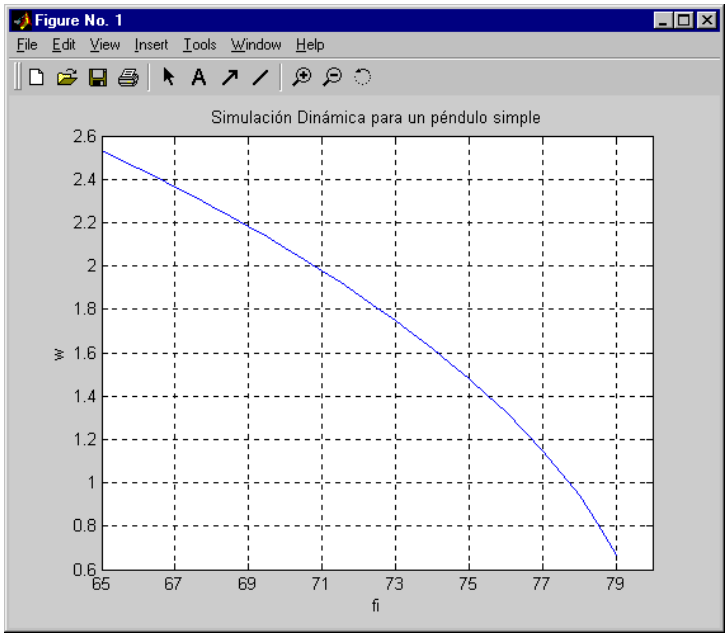
```
function Pendulum()
% Pendulum con Runge-Kutta 4
% Datos
% n   =numero de muestras
% a   =limite inferior de la integral
% b   =limite superior de la integral
% Resultados
% [x y]=ODE por RK4
   clc;
   global L;

   L=input('Ingrese la longitud -> ');%L=2.5
   a=79;a1=(a/180)*pi;% en radianes
   b=65;b1=(b/180)*pi;% en radianes
   n=28;
  fprintf('\n   x en radianes\n   y=velocidad angular\n');
   [x,y]=RK4('dwdfi',a1,b1,n,0.66488546);

   % grafico
   x=(x/pi)*180;

   plot(x,y);
   grid on;
   set(gca,'XTick',b:2:a);
   title('Simulación Dinámica para un péndulo simple');
   xlabel('fi');
   ylabel('w');
```

Archivo *RK4.m*

```
function [x,y]=RK4(f,a,b,n,y0)
%Runge-Kutta 4
% Datos
% f   =el nombre de la funcion como string
% a   =limite inferior
% b   =limite superior
% h   =longitud del segmento
% y0  =f(a);
% x   =es el vector x
% n   =numero de segmentos
% Resultados
% y   =es el vector f(x)
   h=(b-a)/n;
   n=n+1;
   y=zeros(n,1);
   x=zeros(n,1);
   x(1)=a;
   y(1)=y0;
   fprintf('     x         y     \n');
   fprintf('====================\n');
   fprintf('%10.6f%10.6f\n',x(1),y(1));
   for i=1:n-1
      k1=feval(f,x(i),y(i));
      k2=feval(f,x(i)+h/2,y(i)+h*k1/2);
      k3=feval(f,x(i)+h/2,y(i)+h*k2/2);
            k4=feval(f,x(i)+h,y(i)+h*k3);
      x(i+1)=a+h*i;
      y(i+1)=y(i)+h*(k1+2*k2+2*k3+k4)/6;
      fprintf('%10.6f%10.6f\n',x(i+1),y(i+1));
   end
```

Las siguientes figuras presentan la ejecución (para L =2.5) y la gráfica resultante; respectivamente:





6. Un móvil que está en el punto (1,1) y se dirige al punto (2,1) sigue la trayectoria:

$$x\,y'' = 0.5\sqrt{11+(y')^2}$$

a) Aproxime y(x) mediante el método del disparo, con Euler. Use h = 0.25 con una tolerancia de 10^{-4}.

Sol

$$y'' = \frac{0.5}{x}\sqrt{11+(y')^2}$$

C.F. : a=1 y(a)=1 b=2 y(b)=1

$y' = z$ $y(1) = 1$

$$z' = y'' = \frac{0.5}{x}\sqrt{11+(z)^2} = f(x,y,z) \quad z(1) = s_o = 0$$

Algoritmo de Euler

x	y	y'
1	1	s_o=0
1.25	1	0.41458
1.5	1.1036	0.74882
1.75	1.2908	1.0322
2.00	1.5489	1.2803

x	y	y'
1	1	s_1=-0.54889
1.25	0.86278	-0.12867
1.50	0.83061	0.20324
1.75	0.88142	0.48014
2.00	1.0015	0.71951

Interpolando s_2

s_2=-0.55035

x	y	y'
1	1	s_2=-0.55035
1.25	0.86241	-0.13010
1.50	0.82989	0.20182
1.75	0.88034	0.47871
2.00	1.00002	0.71807

b) ¿Cuál es la distancia "d" recorrida por el móvil?

$$d = \int_1^2 \sqrt{1+(y'(x))^2}\,dx = \int_1^2 f(x)dx$$

h=0.25

T=h*0.5*(f_1+f_5+2*sum(f (2:4))) =1.0809

6. Considere el siguiente Problema de Valores de Contorno:

$-u''(x)+u'(x)=x(1-x)$; $x \in (0,1)$

$u(0)=0$ y $u(1)=0$

Considere una partición regular en el intervalo [0,1] con un espaciamiento h = 0.25. Obtener una solución aproximada para el problema de valor frontera usando el método de las diferencias centrales.

Solución:

$$\frac{-u_{i-1}+2u_i-u_{i+1}}{h^2}+\frac{u_{i+1}-u_i}{2h}=x_i(1-x_i)$$

$$(-1-\frac{h}{2})u_i-1+2u_i+(-1+\frac{h}{2})u_{i+1}=h^2x_i(1-x_i) \qquad \text{h=0.25}$$

N+1=(1-0)/h

N+1=4 N=3

i =1,2,3

(-1 - h/2)=-1.125

(-1+h/2)=-0.875

$$\begin{bmatrix} 2 & -0.875 & 0 \\ -1.125 & 2 & -0.875 \\ 0 & -1.125 & 2 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}=\begin{bmatrix} h^2(0.1875) \\ h^2 0.25 \\ h^2(0.1875) \end{bmatrix}$$

Solución

$$y=\begin{bmatrix} 0.0176 \\ 0.0269 \\ 0.0210 \end{bmatrix}$$

Programa en MATLAB

```
% p.m
function f=p(t)
  f=1;
```

```
% q.m
function f=q(t)
  f=0;
```

```
% r.m
function f=r(t)
  f=t*(t-1);
```

```
% trisys.m
function X = trisys(A,D,C,B)
%-------------------------------------------------------
% Solucion del sistema tridiagonal
% Entradas
%   A   vector sub diagonal
%   D   vector diagonal
%   C   vector super diagonal
%   B   vector del lado derecho
% Salida
%   X   vector solucion
%-------------------------------------------------------
n = length(B);
for k = 2:n,
  mult = A(k-1)/D(k-1);
  D(k) = D(k) - mult*C(k-1);
  B(k) = B(k) - mult*B(k-1);
end
X(n) = B(n)/D(n);
for k = (n-1):-1:1,
  X(k) = (B(k) - C(k)*X(k+1))/D(k);
end
```

```
% findiff.m
function [T,X] = findiff(p,q,r,a,b,alpha,beta,n)
%-------------------------------------------------------
% Solucion del problema de valor de frontera usando
% diferencias finitas
%         x(t)'' = p(t)x'(t)+q(t)x(t)+r(t)
%         x(a) = alpha, x(b) = beta
% Entradas
%   p,q,r   Nombres de las funciones
%   a,b     Limites del intervalo [a,b]
%   alpha   Valor frontera izquierdo
%   beta    Valor frontera derecho
%   n       numero de pasos
% Salida
%   T       Vector de abscisas
%   X       Vector de ordenadas
%-------------------------------------------------------
T  = zeros(1,n+1);   X  = zeros(1,n-1);
```

```
Va = zeros(1,n-2);   Vb = zeros(1,n-1);
Vc = zeros(1,n-2);   Vd = zeros(1,n-1);
h = (b - a)/n;
for j=1:n-1,
  Vt(j) = a + h*j;
end
for j=1:n-1,
  Vb(j) = -h^2*feval(r,Vt(j));
end
Vb(1)   = Vb(1)   + (1 + h/2*feval(p,Vt(1)))*alpha;
Vb(n-1) = Vb(n-1) + (1 - h/2*feval(p,Vt(n-1)))*beta;
for j=1:n-1,
  Vd(j) = 2 + h^2*feval(q,Vt(j));
end
for j=1:n-2,
  Va(j) = -1 - h/2*feval(p,Vt(j+1));
end
for j=1:n-2,
  Vc(j) = -1 + h/2*feval(p,Vt(j));
end
X = trisys(Va,Vd,Vc,Vb);
T = [a,Vt,b];
X = [alpha,X,beta];
» [T,X] = findiff('p','q','r',0,1,0,0,4)

T =        0    0.2500    0.5000    0.7500    1.0000
X =        0    0.0176    0.0269    0.0210         0
```

Metodo del disparo

El problema de valor frontera se reduce a uno de valor inicial, es un metodo de prueba y error donde se tanteando la pendiente en el punto inicial.

```
% Disparo.m
% y''-y'-2y=0
% y(0)=0.1
% y(0.5)=0.283
% h=0.1
x0=0,y0=0.1,b=0.5,B=0.283
S0=(B-y0)/(b-x0)
y=rk4s('fu2',0,0.5,[0.1 S0],0.1)
ynS0=y(6,2)
plot(y(:,1),y(:,2)), grid, hold on
S1=S0+(B-ynS0)/(b-x0)
y=rk4s('fu2',0,0.5,[0.1 S1],0.1)
ynS1=y(6,2)
plot(y(:,1),y(:,2)), grid
S2=S0+(S1-S0)*(B-ynS0)/(ynS1-ynS0)
y=rk4s('fu2',0,0.5,[0.1 S2],0.1)
```

```
ynS1=y(6,2)
plot(y(:,1),y(:,2)), grid
err=abs(ynS1-B)
hold off

% x0 =     0
% y0 =     0.1000
%  b =     0.5000
%  B =     0.2830
% S0 =     0.3660
% y =
%       0    0.1000    0.3660
%   0.1000    0.1397    0.4295
%   0.2000    0.1864    0.5088
%   0.3000    0.2420    0.6071
%   0.4000    0.3086    0.7285
%   0.5000    0.3887    0.8780
%
% ynS0 =   0.3887
% S1   =   0.1547
%
% y =
%       0    0.1000    0.1547
%   0.1000    0.1174    0.1937
%   0.2000    0.1390    0.2409
%   0.3000    0.1659    0.2981
%   0.4000    0.1990    0.3677
%   0.5000    0.2399    0.4523
%
% ynS1  =    0.2399
% S2    =    0.2159
%
% y =
%       0    0.1000    0.2159
%   0.1000    0.1238    0.2620
%   0.2000    0.1527    0.3185
%   0.3000    0.1879    0.3876
%   0.4000    0.2308    0.4722
%   0.5000    0.2830    0.5756
% ynS1 = 0.2830
% err  = 5.5511e-017
```

5. Ejercicios Propuestos

1. Sea el problema de condición inicial

$$\begin{cases} y'(t) = \dfrac{1}{1+y^2} \\ y(0) = 1 \end{cases}$$

Use el método de Taylor de orden 2 para determinar el valor aproximado de y(1), con tamaño de paso h=0.5. Justifique cada paso

……
……
……

2. Probar que la función $f(t,y) = t|y|$ es Lipschitziana , con respecto a la variable *y*, en el conjunto: $D = \{(t,y) \in R^2 / \quad 1 \le t \le 2; \quad -3 \le y \le 4\}$

……
……
……

3. Considere la ecuación diferencial $y''+4ty'+2y^2 = 0$ con condiciones iniciales $y(0) = 1, \quad y'(0) = 0$ con h=0.1, utilice el método de Euler para aproximar y(0.2) e y'(0.2)

……
……
……
……

4. Sea la ecuación de Blasius:

$y''' = -y\, y''$

Con las condiciones iniciales: $y(0) = 1 \quad y'(0) = 2 \quad y''(0) = 0$, obtener $y(0.2)$ usando Euler con $h = 0.1$.

$y(0.2)$=.....

5. Para la EDO : $2\dfrac{dy}{dx} + 3y = e^{-5x}, y(0) = 7$, encuentre la constante L de Lipschitz del teorema de existencia y unicidad de la EDO.
L= ……..

6. Considere la ecuación diferencial $y''+4ty'+2y^2 = 0$ con condiciones iniciales $y(0) = 1, \quad y'(0) = 0$ con h=0.1, utilice el método de Euler para aproximar y(0.2) e y'(0.2).

……
……
……
……

7. Considere el problema de valor inicial
$y' = (y - x - 1)^2 + 2$
$y(0) = 1$
a) Muestre que $y(x) = 1 + x + \tan x$ es solución exacta del problema dado

b) Obtener una aproximación para y(0.1) usando Runge Kutta de orden 2 con h=0.1

8. Un sistema dinámico obedece a la siguiente ecuación diferencial ordinaria:

$$\ddot{\theta} - \frac{g}{l}\theta = 0$$
$$\theta(0) = 0\ rad$$
$$\dot{\theta}(0) = 0.25\ rad/seg$$

Si $g = 9.81\ m/seg^2$, y $l = 0.5\ m.$; estime $\theta(0.15)$, usando el método de Euler (h=0.05).

9. Se lanza un cuerpo de masa constante *m* hacia arriba; desde la superficie terrestre, con una velocidad inicial v_o. Encontrar la ODE para la velocidad; suponiendo que no existe resistencia del aire.

Sugerencia: Desde la Ley de Newton, el peso de un cuerpo de masa *m* es el cuadrado inverso de la atracción de la gravedad. Si *R* es el radio de la tierra y ***x*** la altura desde la superficie; el peso viene dado por la expresión:

$$w(x) = \frac{K}{(x+R)^2}$$

Observe: que $w(0) = mg$

10. Un piloto con su paracaídas, salta desde un aeroplano. El peso combinado del piloto y sus paracaídas es *W*. Suponiendo que existe una fuerza que se opone al movimiento; siendo proporcional a la velocidad de la caída. Presentar la ODE, resolverla y analizar la velocidad crítica.

	Curso	**Métodos Numéricos**	**Código : MB536**
	Tema	**Ecuaciones Diferenciales Parciales**	
	Practica	**09**	
	Profesores	**Garrido Juárez, Rosa**	**Castro Salguero, Robert**
		Hermes Pantoja, Carhuavilca	**Obregón Ramos, Máximo**
		Ruiz Lizama, Edgar	

1. Objetivos

Estudiar y aplicar los métodos para resolver numéricamente ecuaciones diferenciales parciales.

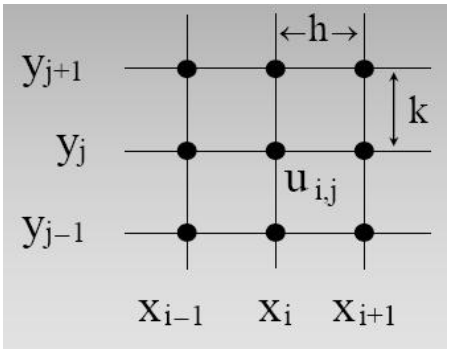
2. Fundamento Teórico

Una ecuación diferencial parcial (EDP) puede ser escrita en forma general

$$a\frac{\partial^2\phi}{\partial x^2}+b\frac{\partial^2\phi}{\partial x\partial y}+c\frac{\partial^2\phi}{\partial y^2}+d\frac{\partial\phi}{\partial x}+e\frac{\partial\phi}{\partial y}+f\phi+g=0$$

donde a,b,c,d,e,f y g puede ser funciones de variables independientes x e y , también de variables dependientes ϕ, en una región $\mathscr{R}$ del plano cartesiano.
Las EDPs pueden ser clasificadas en Elípticas, Parabólicas o Hiperbólica,

- si $b^2 - 4ac < 0$, la ecuación es llamada **Elíptica** .
- si $b^2 - 4ac = 0$, la ecuación es llamada **Parabólica** .
- si $b^2 - 4ac > 0$, la ecuación es llamada **Hiperbólica** .
- Discretización: EDP ➔ EDF
- Métodos explícitos
  - Sencillos
  - Inestables
- Métodos implícitos
  - Más complejos
  - Estables



2.1 EDPs Elípticas

Podemos citar a la *ecuación de Poisson*

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=g(x,y) \qquad (1)$$

o de Laplace

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=0$$

2.1.1 Método Explicito

Para la ecuación (1) aproximaremos la segunda derivada a través de la formula de diferencia finita central

$$\frac{\partial^2 u}{\partial x^2}\approx\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2}$$

$$\frac{\partial^2 u}{\partial y^2}\approx\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{k^2}$$

donde *h* e *k* son los espaciamientos en las direcciones de *x* e *y*, respectivamente.

Reemplazando en (1), obtenemos

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = g(x, y)$$

Estas ecuaciones, con las condiciones de frontera dan un sistema lineal con (n-1)(m-1) incógnitas. Este sistema podría ser resuelto por eliminación Gaussiana (u otros métodos directos) o métodos iterativos como Gauss-Seidel.

Las *condiciones de borde o de frontera* deben estar especificadas para que exista una solución única. Especificar el valor de la función en el borde es la forma más simple y se la conoce como *condición de frontera de Dirichlet* o *condición forzada.*

2.2 EDPs Parabólicas

Sea la ecuación unidimensional:

$$\frac{\partial^2 U}{\partial x^2} = \frac{\partial U}{\partial t}$$

2.2.1 Método Explicito

Para la segunda derivada respecto de la variable *x*, podemos hacerla con una diferencia dividida central con una aproximación de segundo orden:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

y una diferencia dividida finita hacia delante para aproximar a la derivada en el tiempo:

$$\frac{\partial u}{\partial t} \approx \frac{u_{i+1,j} - u_{i,j}}{k}$$

Sustituyendo estas aproximaciones:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} = \frac{u_{i+1,j} - u_{i,j}}{k}$$

Despejando:

$$u_{i+1,j} = ru_{i,j-1} + (1 - 2r)u_{i,j} + ru_{i,j+1}, \quad r = \frac{k}{h^2}$$

El cual nos da la temperatura U en cada punto j en (i + 1)-iésimo tiempo. Note que los puntos discretos son xj = jh y ti = ik

2.3 EDPs Hiperbólicas

La ecuación a tratar en esta oportunidad es la ecuación de la onda unidimensional, cuya expresión es:

$$\frac{\partial^2 u}{\partial x^2} = c\frac{\partial^2 u}{\partial t^2} \qquad 0 < x < L, \qquad t > 0$$

Condiciones Iniciales

$u(x, 0) = f(x)$ $\quad u_t(x, 0) = g(x)$

Condiciones de Contorno

$u(0,t) = l(t)$ $\qquad u(L,t) = r(t)$

2.3.1 Método Explicito

$$\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = c^2 \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

Condiciones iniciales

$u_{i,0} = f_i$ y $u_{i,1} - u_{i,-1} = 2kg_i$

Paso 1º

$u_{i,1} = \alpha^2(f_{i-1}+f_{i+1})/2 + (1-\alpha^2)f_i + kg_i$

Pasos siguientes

$u_{i,j-1} = \alpha^2 (u_{i+1,j} + u_{i-1,j}) + 2(1 - \alpha^2)u_{i,j} - u_{i,j-1}$

Convergencia $\alpha < 1$

3. Instrucciones básicas en MATLAB

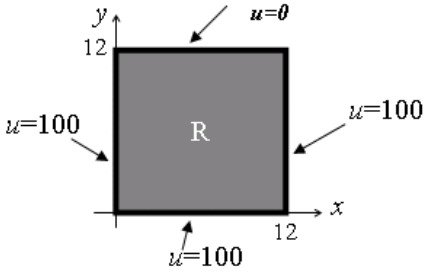
```
% u(x,0)= f1(x)
function f=f1(x)
f=0;

% u(x,b)= f2(x)
function f=f2(x)
f=200*x;

% u(0,y)= f3(y)
function f=f3(y)
f=0;

% u(a,y)= f4(y)
function f=f4(y)
f=200*y;

function U = dirich(f1,f2,f3,f4,a,b,h,tol,max1)
%DIRICH   Dirichlet solution to Laplace's equation.
% Sample call
%   U = dirich('f1','f2','f3','f4',a,b,h,tol,max1)
% Inputs
%   f1     name of a boundary function
%   f2     name of a boundary function
%   f3     name of a boundary function
%   f4     name of a boundary function
%   a      width of interval [0 a]: 0<=x<=a
%   b      width of interval [0 b]: 0<=y<=b
%   h      step size
%   tol    convergence tolerance
%   max1   maximum number of iterations
% Return
%   U      solution: matrix

n = fix(a/h)+1;
m = fix(b/h)+1;
ave = (a*(feval(f1,0)+feval(f2,0)) + b*(feval(f3,0)+feval(f4,0)))/(2*a+2*b);
```

```
U = ave*ones(n,m);
for j=1:m,
U(1,j) = feval(f3,h*(j-1));
U(n,j) = feval(f4,h*(j-1));
end
for i=1:n,
U(i,1) = feval(f1,h*(i-1));
U(i,m) = feval(f2,h*(i-1));
end
U(1,1) = (U(1,2) + U(2,1))/2;
U(1,m) = (U(1,m-1) + U(2,m))/2;
U(n,1) = (U(n-1,1) + U(n,2))/2;
U(n,m) = (U(n-1,m) + U(n,m-1))/2;
w = 4/(2+sqrt(4-(cos(pi/(n-1))+cos(pi/(m-1)))^2));
err = 1;
cnt = 0;
while ((err>tol)&(cnt<=max1))
err = 0;
for j=2:(m-1),
    for i=2:(n-1),
    relx = w*(U(i,j+1)+U(i,j-1)+U(i+1,j)+ U(i-1,j)-4*U(i,j))/4;
    U(i,j) = U(i,j) + relx;
    if (err<=abs(relx)), err=abs(relx); end
    end
end
cnt = cnt+1;
end

>>U = dirich('f1','f2','f3','f4',0.5,0.5,0.125,1e-6,100)

U =

       0        0        0        0   12.5000

       0    6.2500   12.5000   18.7500   25.0000
       0   12.5000   25.0000   37.5000   50.0000
       0   18.7500   37.5000   56.2500   75.0000

  12.5000   25.0000   50.0000   75.0000   75.0000
```

4. Parte práctica

Ejemplo 1:

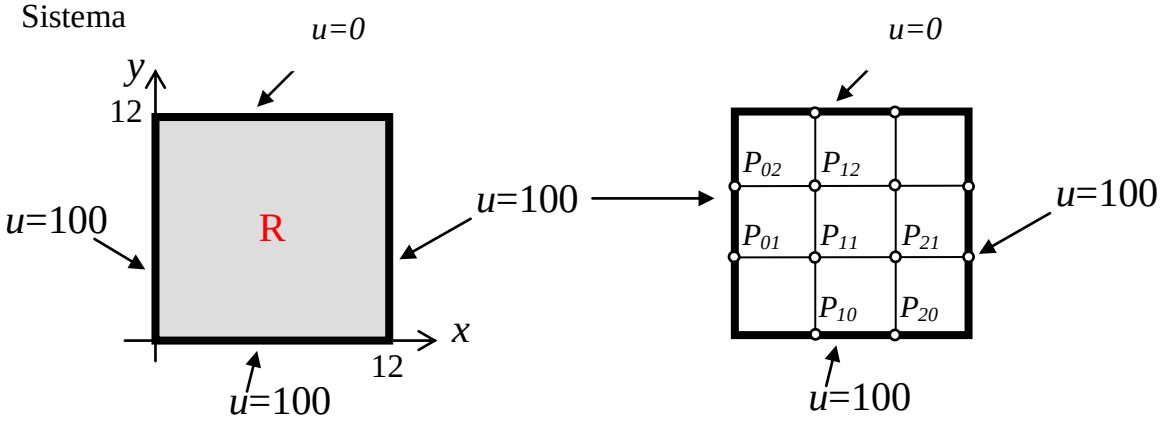
Una placa de 12 cm de lado tiene sus borde mantenidos a las temperaturas mostradas en la figura. Se desea saber la distribución de temperatura en el interior de la placa. Se escogerá un espaciamiento de h=4cm.



La ecuación de transferencia de calor en estado estacionario se reduce a Laplace $\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$. Plantee el sistema lineal en los nodos pedidos.

Solución

Sistema



Nodo P_{11} $100+P_{12}+P_{21}+100-4P_{11}=0$ Nodo P_{21} $P_{11}+P_{13}+100+100-4P_{21}=0$ Nodo P_{12} $100+0+P_{13}+P_{11}-4P_{12}=0$ Nodo P_{13} $0+P_{12}+100+P_{21}-4P_{13}=0$	$\begin{bmatrix}-4 & 1 & 1 & 0\\ 1 & -4 & 0 & 1\\ 1 & 0 & -4 & 1\\ 0 & 1 & 1 & -4\end{bmatrix}\begin{bmatrix}P_{11}\\P_{21}\\P_{12}\\P_{13}\end{bmatrix}=\begin{bmatrix}-200\\-200\\-100\\-100\end{bmatrix}$

Ejemplo 2:

Resolver el siguiente EDP

$u_{tt} = c^2 u_{xx}$, $0 < x < L$, $t > 0$

$c = 1$, $L=T=4$, $n_x=4$, $n_t=8$,

$u(x, 0) = 2-|x-2|$ $u_t(x, 0) = 0$
$u(0,t) = 0$ $u(L,t) = 0$

Instante t = 0:
$u_{0,0} = f(x_0) = 2 - |x_0 - 2| = 2 - |0 - 2| = 0 = f(x_4)$
$u_{1,0} = f(x_1) = 2 - |x_1 - 2| = 2 - |1 - 2| = 1 = f(x_3)$
$u_{2,0} = f(x_2) = 2 - |x_2 - 2| = 2 - |2 - 2| = 2$

Instante t=1:
$u_{i,1} = a_2 \cdot (u_{i-1,0}+u_{i+1,0})/2 + (1 - a_2) \cdot u_{i,0} + k \cdot g(x_i)$
donde $a_2 = ¼$, $1 - a_2 = ¾$:
$u_{1,1} = (1/4)(u_{0,0} + u_{2,0})/2 + (3/4)u_{1,0} =$
$(1/4)(0 + 2)/2 + (3/4)1 = 1 = u_{3,1}$
$u_{2,1} = (1/4)(u_{1,0} + u_{3,0})/2 + (3/4)u_{2,0} =$
$(1/4)(1 + 1)/2 + (3/4)2 = 7/4$

Aplicando la fórmula genérica
$u_{i,j+1} = a_2 \cdot (u_{i-1,j} + u_{i+1,j}) + 2 \cdot (1 - a_2) \cdot u_{i,j} - u_{i,j-1}$
con lo que, para t = 1 obtenemos:
$u_{1,2} = (1/4)(u_{0,1} + u_{2,1}) + (3/2)u_{1,1} - u_{1,0}$
$= (1/4)(0 + 7/4) + (3/2)1 - 1 = 15/16$
$= u_{3,2}$
$u_{2,2} = (1/4)(u_{1,1} + u_{3,1}) + (3/2)u_{2,1} - u_{2,0}$
$= (1/4)(1 + 1) + (3/2)(7/4) - 2 = 9/8$

Ejemplo 2

La distribución de temperaturas en una placa rectangular de 1m. x 0.9m. obedece a la siguiente ecuación diferencial parcial:

$$xy\frac{\partial^2 T}{\partial x^2} + (x + y)\frac{\partial^2 T}{\partial y^2} = 2x + y \quad 0 \le x \le 1 \quad 0 \le y \le 0.9$$

Se sabe que en el borde exterior la temperatura obedece a la siguiente relación:

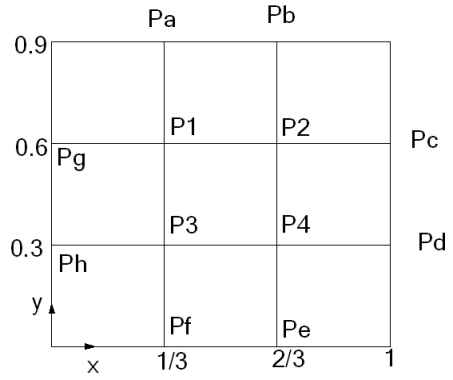
$$T(x, y) = x^2 + y^2$$

Se desea hallar la distribución interna de temperaturas.

a) Realice la discretizacion usando una malla rectangular (Δx =1/3, Δy =0.3)
b) Plantee el sistema de ecuaciones lineales usando el método de las diferencias finitas

Solución

a)



c)

Condiciones de Frontera:

$$P_a = (1/3)^2 + 0.9^2$$
$$P_b = (2/3)^2 + 0.9^2$$
$$P_c = (1)^2 + 0.6^2$$
$$P_d = (1)^2 + 0.3^2$$
$$P_e = (2/3)^2 + 0^2$$
$$P_f = (1/3)^2 + 0^2$$
$$P_g = (0)^2 + 0.6^2$$
$$P_h = (0)^2 + 0.3^2$$

b) Planteando las ecuaciones en diferencias finitas:

Nodo P1: x=1/3 y=0.6

$$\frac{1}{3}(0.6)\left(\frac{P_2 - 2P_1 + P_g}{\Delta x^2}\right) + \left(\frac{1}{3} + 0.6\right)\left(\frac{P_a - 2P_1 + P_3}{\Delta y^2}\right) = 2\left(\frac{1}{3}\right) + 0.6$$

Nodo P2 x=2/3 y=0.6

$$\frac{2}{3}(0.6)\left(\frac{P_c - 2P_2 + P_1}{\Delta x^2}\right) + \left(\frac{2}{3} + 0.6\right)\left(\frac{P_b - 2P_2 + P_4}{\Delta y^2}\right) = 2\left(\frac{2}{3}\right) + 0.6$$

Nodo P3 x=1/3 y=0.3

$$\frac{1}{3}(0.3)\left(\frac{P_4 - 2P_3 + P_h}{\Delta x^2}\right) + \left(\frac{1}{3} + 0.3\right)\left(\frac{P_1 - 2P_3 + P_f}{\Delta y^2}\right) = 2\left(\frac{1}{3}\right) + 0.3$$

Nodo P4 x=2/3 y=0.3

$$\frac{2}{3}(0.3)\left(\frac{P_d - 2P_4 + P_3}{\Delta x^2}\right) + \left(\frac{2}{3} + 0.3\right)\left(\frac{P_2 - 2P_4 + P_e}{\Delta y^2}\right) = 2\left(\frac{2}{3}\right) + 0.3$$

5. Ejercicios Propuestos

1. Dada la E.D.P

$$(x+1)\frac{\partial^2 w}{\partial x^2} + (y^2+1)\frac{\partial^2 w}{\partial y^2} - w = 1$$

$$0 \le x \le 1, \quad 0 \le y \le 1, \quad \Delta x = \Delta y = 1/3$$

Con las condiciones de frontera:

$$w(0, y) = y$$
$$w(1, y) = y^2$$
$$w(x, 0) = 0$$
$$w(x, 1) = 1$$

a) Dibuje la malla con las incógnitas y los valores frontera
b) Plantear el sistema de ecuaciones mediante diferencias finitas

2. Determine el sistema de cuatro ecuaciones con las incógnitas p_1, p_2, p_3 y p_4 para calcular la aproximación de la función armónica u(x,y) en el rectángulo R={(x,y): $0 \le x \le 3$, $0 \le y \le 3$}. Los valores de frontera son:

u(x,0) = 10 y u(x,3)=90 para $0<x<3$
u(0,y) = 70 y u(3,y)=0 para $0<y<3$