	Curso	Cálculo Numérico	Código : MB535
	Tema	Introducción al Matlab	
	Practica	01	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

1. Objetivos :

Manejar los comandos básicos para operaciones con vectores y matrices.

2. Fundamento Teórico

Se puede utilizar MATLAB como simple calculadora, escribiendo expresiones aritméticas y terminando con **[enter]**. 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 ($3+4-5=7-5$)

Números y Formatos

Matlab reorganiza diferentes clases de números

Tipo	Ejemplo
Entero	1362, -217897
Real	1.234, -10.76
Complejo	$3.21 - 4.3 i$ ($i = \sqrt{-1}$)
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:

$$-1.3412e+03 = -1.3412 \times 10^3 = -1341.2$$

$$-1.3412e-01 = -1.3412 \times 10^{-1} = -0.13412$$

Operadores relacionales:

<	Menor
<=	Menor o igual
>	Mayor
>=	Mayor o igual
==	Igual
~=	Diferente

Operadores para matrices

\wedge * / + -	Potenciación , producto, división, suma, resta matricial
\cdot \wedge .* ./ \	Producto y división elemento a elemento. $A \cdot B = B \cdot A$
/	División matricial por la derecha: Si $C=A * B \rightarrow C/B=A=C*inv(B)$
\	División matricial por la izquierda: Si $C=A * B \rightarrow A \setminus C=B=inv(A)*C$
'	Traspuesta

3. Instrucciones básicas en Matlab

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	1.33333333333333e+000 1.234499317939127e-006
format hex	3ff5555555555555 3eb4b6225ac42812
format bank	1.33 0.00
format rat	4/3 1/810045

Funciones matemáticas elementales			
Comando	Descripción	Comando	Descripción
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

Funciones de propósito general	
Comando	Descripción
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

Selección de los elementos de un vector x o matriz A	
Función	Descripción
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(3:end,:)	Devuelve desde la tercera hasta la última fila
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
diag(V)	Si V es un vector, devuelve una matriz cuadrada donde V se ubica en la diagonal de dicha matriz, los demás elementos serán 0
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
max(A),min(A)	Devuelve un vector de los máximos o mínimos de cada columna
sum(A)	Devuelve un vector de la suma de cada columna
[n,m]=size(A)	Devuelve el número de filas y columnas de la matriz A

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.

Funciones para gráficos	
Función	Descripción
ezplot('expresión de la función')	dibuja la gráfica de la función dada por la expresión, para x variando en un intervalo por defecto
ezplot('expresión de la función',[a,b])	dibuja la gráfica de la función dada por la expresión, para x variando en el intervalo [a,b]
figure(n)	Crea o activa una ventana donde se puede realizar un gráfico, por defecto la gráfica se realiza en la ventana 1.
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 versus los índices de 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
linspace(a,b,np)	Devuelve un vector de np valores desde a hasta b espaciados igualmente.

4. Parte práctica

Instrucciones básicas

```
>> 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)

>> (3+4^2)/(2/(3^(1/5))-(1/(3.1-2))^(3/4))
```

Resuelve $\frac{3+4^2}{\sqrt[5]{3}-\left(\frac{1}{3.1-2}\right)^{3/4}}$

Instrucciones para matrices

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

Resultaría en la matriz

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

```
>>x = [-1.3,sqrt(3),(1+2+3) *4/5]
```

Resultaría en

```
x =
-1.3000 1.7321 4.8000
```

```
>>x(5) = abs(x(1))
```

Resultaría en

```
x =
-1.3000 1.7321 4.8000 0 1.3000
```

Para añadir otra fila a la matriz A de arriba podemos hacer lo siguiente:

```
>>r = [10 11 12];
```

```
>>A = [A; r]
```

Resultaría en

```
A =
     1 2 3
     4 5 6
     7 8 9
    10 11 12
```

```
>>x = [1 2 3]; y = [4 5 6];
```

```
>>z = x. *y
```

Resulta en

```
z =
     4 10 18
>>x = 1:5
```

genera un vector fila que contiene los números enteros del 1 al 5:

```
x =
     1 2 3 4 5
```

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

```
>>A(3, 3) = A(1, 3) + A(3, 1)
```

Por ejemplo, suponga que A es una matriz 10 por 10. Entonces

>>A(1:5, 3) Especifica la submatriz 5 x 1, ó vector columna, que consiste de los primeros cinco elementos en la tercera columna de A.

>>A(1:5, 7:10)Es la submatriz 5 x 4 de las primeras cinco filas y las últimas cuatro columnas. Utilizando solo los dos puntos denota *todo* lo correspondiente a la fila ó columna.

>>B=A;

>>A(:, [3 5 10]) = B(:, 1:3) Reemplaza la tercera, quinta y décima columna de A con las primeras tres columnas de B.

Gráficos

>> ezplot('sin(x^2)*x/2') dibuja la gráfica de la función

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

Grafica de la campana de gauss

>>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

5. Ejercicios Propuestos

5.1 Evaluar las siguientes expresiones matemáticas en MATLAB.

a. $\log_{10}(2)$

c. $\sqrt{5 + e^2}$

b. $|\arcsen(-0.5)|$

d. $\tan(e)$

e) $\frac{1}{\frac{2}{0.1^{1/2}} - \frac{0.4}{2^{1/3}}}$

5.2 Extraer las siguientes submatrices de A=rand(20,10):

a. Las 5 primeras columnas

b. Las 15 ultimas columnas

c. primera y quinta fila

d. 10 últimos elementos de la tercera fila

e. La intersección de la fila:2,3,17,19 con las columnas:5 6 7 12

5.3 Crear los siguientes vectores:

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

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

5.4 Crear un vector z de cuatro números complejos

5.5 Listar el tercer elemento del vector z

5.6 Listar los 5 primeros elementos del vector y

5.7 Listar los 5 últimos elementos del vector y

5.8 Listar los elementos de posiciones impares del vector y

5.9 Listar los elementos de posiciones 2, 4, 5, y 7 del vector y

5.10 Crear los vectores a = [1 2 3 4 5] y b = [1 3 5 7 9]

5.11 Fusionar los vectores a y b en un vector c

5.12 Obtener la transpuesta del vector c

5.13 Obtener la transpuesta del vector z

5.14 Crear las siguientes matrices: $g = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$ $h = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix}$

5.15 Sumar las matrices g y h

5.16 Multiplicar las matrices g y h

5.17 Multiplicar g con la transpuesta de h

5.18 Multiplique g y h componente a componente

5.19 Eleve 2 a cada elemento de g

5.20 Obtener la inversa de cada elemento de g.

5.21 Resolver el sistema:

$$2a + 3b + c = 6$$

$$4a + b + 2c = 7$$

$$6a + b + 7c = 4$$

Mediante la función **inv**.

5.22 Resuelva el sistema anterior mediante \.

5.23 Utilizando MATLAB determine el valor de la expresión

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

5.24 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

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

5.25 Escriba las matrices A y B definidas por

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

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

5.26 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**.

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

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

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

5.29 Dada la matriz

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

Que operaciones se utilizaron para obtener las siguientes matrices:

(a)	ans =	7	8	9
		4	5	6
		1	2	3
(b)	ans =	8	7	9
		5	4	6
		2	1	3
(c)	A =	1	3	
		7	9	

5.30 Graficar la siguiente función

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

5.31 Graficar la función

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

	Curso	Cálculo Numérico	Código : MB535
	Tema	Programación en Matlab y Teoría de Errores	
	Practica	02	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

1. Objetivos

- Aplicar las instrucciones de control básicas en Matlab, para la implementación de programas.
- Estudio del comportamiento de los errores y notación en punto flotante.

2. Fundamento Teórico

2.1 Archivos *.m

Un archivo *.m es un archivo con un conjunto de comandos que Matlab puede interpretar.

Distinguiremos dos tipos de archivos *.m dentro del Matlab:

Archivo script: es un archivo con un conjunto de comandos que Matlab ejecuta siempre de la misma manera y que no necesita ninguna variable de entrada y no proporciona ninguna variable de salida.

Archivo tipo Función: es un archivo con un conjunto de comandos que necesita una ó más variables de entrada para que Matlab pueda ejecutar. Puede tener variables de salida. Todos los archivos *.m que sean funciones empiezan:

function [salida1, salida2,...] =nombre(entrada1, entrada2,...);

siendo entrada1, entrada2 ,..... Parámetros necesarios para la función y

salida1, salida2, Parámetros opcionales de salida.

2.2 Creación de un Archivos 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:
 - i. **cd c:\MB535_perez** y presione la tecla **Enter**
- c) Crear un nuevo archivo-m:
 - a. 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:

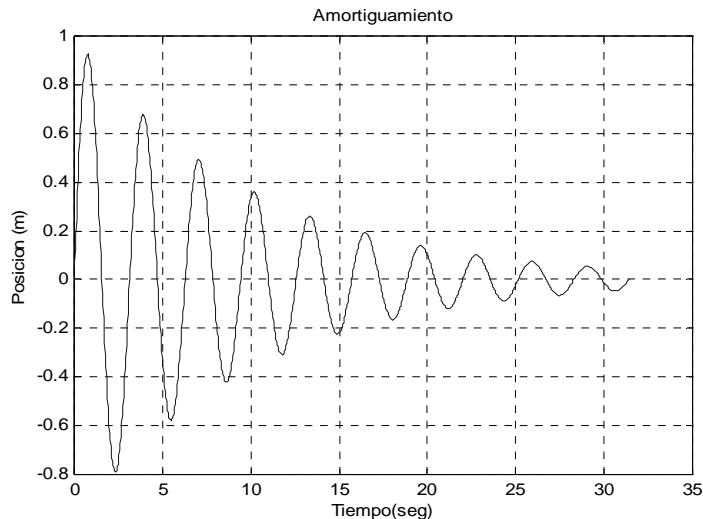
En la ventana de comandos escriba el nombre del programa: **prueba01** y presione la tecla **Enter**.

El programa solicitará el ingreso de un dato:

Ingrese número de períodos =

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

Se mostrará el siguiente gráfico:



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

2.3 Funciones

El Matlab permite la creación de funciones definidas por el programador, la cual contiene una cabecera como se indica abajo y debe ser guardado con extensión **m**.

```
function [argumentos salida] = nombre_ función (argumentos entrada)  
<instrucciones ejecutables>  
Asignar valor a los argumentos de salida
```

[argumentos de salida] : si es más de un argumento se separan por comas.

si es un argumento se puede omitir los []

(argumentos de entrada) : si es más de uno separado por comas.

Las funciones se deben grabar como M-files.

Teoría de Errores

2.4 Tipos de errores:

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

2.5 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.

2.6 Propagación de Errores

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

$$\xi_y \approx |y'| \xi_x$$

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

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

2.7 Errores de redondeo y aritmética de punto flotante

Los números se almacenan en la computadora como una secuencia de dígitos binarios o *bits* (unos o ceros), pero para analizar los efectos de los errores de redondeo, se supone que los números se representan en la *forma normalizada decimal de punto flotante*:

$$\pm d_0.d_1d_2d_3 \dots d_m \times 10^n, \text{ con } 1 \leq d_0 \leq 9, 0 \leq d_i \leq 9 \text{ para } i = 1, 2, 3, \dots, m$$

La secuencia de dígitos $d_0, d_1, d_2, \dots, d_m$ se conoce como la *mantisa* y el índice n como el *exponente*.

El manejo finito que hace el computador de los números implica que existe un número máximo, digamos k , de dígitos por medio del cual puede representarse un valor; esto es, la mantisa sólo debe contener k dígitos.

Error de redondeo: se ocupa de las operaciones aritméticas en punto flotante tales como la suma, substracción, multiplicación, y división.

- *Punto flotante de precisión* (ϵ es la precisión de la maquina: si $1 < x < 1 + \epsilon$, entonces $x = 1$)
- *Punto flotante underflow* (x_{\min} es el cero de la maquina: si $0 \leq x < x_{\min}$, entonces $x = 0$)
- *Punto flotante overflow* (x_{\max} es el infinito de la maquina: si $x > x_{\max}$, entonces $x = \text{inf}$)

3. Instrucciones básicas en Matlab

3.1 Controles de Flujo y Sentencia de Decisión

3.1.1 La sentencia IF

Condicional simple	Condicional doble	Condicional múltiple
<pre> if condicion sentencias end </pre>	<pre> if condicion sentencias1 else sentencias2 end </pre>	<pre> if condicion1 sentencias1 elseif condicion2 sentencias2 elseif condicion3 sentencias3 else sentenciasN end </pre>

3.1.2 La Sentencia SWITCH

```

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

```

3.2 Operadores lógicos y relacionales

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

3.3 Las Sentencias FOR y WHILE

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

3.4 La sentencias BREAK

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

3.5 Entrada y Salida en un archivo script

Salida:

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

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

error...... Visualiza texto en caso de error y el 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 %3.1fn',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 %3.1fn',,var3,var4,var1,var2)

Entrada:

InputPermite la entrada de valores desde el teclado y se asigna en variables

Conversión Simbólica y Numérica –Errores

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'

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

retorna una representación simbólica de punto flotante

```
'1.9999999999999a'*2^(-4)
```

Los trece dígitos hexadecimales luego del punto decimal corresponden a la mantisa.

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 $1023+e=1023+(-4)=1019=3 \times 16^2 + 15 \times 16 + 11 = 3fb_{(16)}$

```
» format short % formato por defecto
```

La opción 'r'

```
>>sym(t, 'r' )  
retorna la forma racional  
1/10
```

Ésta es la forma fijada por defecto para sym.. Es decir, la llamada a sym sin un segundo argumento es igual que usar sym con la opción 'r':

```
>>sym(t)
```

```
ans =  
1/10
```

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

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

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

La cuarta opción 'd' vuelve la extensión decimal de t hasta el número de dígitos significativos especificados por digits

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

```
ans =  
.10000000000000000000000555111512312578
```

El valor prefijado de `digits` 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
```

Un uso particularmente eficaz de `sym` es convertir una matriz numérica a la forma simbólica. El comando

```
>>A = hilb(3)
```

genera la matriz de Hilbert de 3-x-3

```
»      ans =

      1.0000      0.5000      0.3333
      0.5000      0.3333      0.2500
      0.3333      0.2500      0.2000
```

Aplicando `sym` a A

```
» A = sym(A)
```

```
» A =
```

```
[ 1, 1/2, 1/3 ]
[ 1/2, 1/3, 1/4 ]
[ 1/3, 1/4, 1/5 ]
```

Precisión aritmética de las variables (VPA)

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.

```
vpa(A,4)
```

```
»      A =

      [1.100, 1.200, 1.300]
      [2.100, 2.200, 2.300]
      [3.100, 3.200, 3.300]
```

Convirtiendo de simbólico a punto flotante

Para convertir un racional o variable a su representación de punto flotante en MATLAB, use la función **double**.

4. Parte práctica

4.1 Programación:

Ejemplo 1

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:

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

```
% 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_x1_x2_..','x1','x2',..)
```

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
```

Funciones recursivas

El 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 2 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 3 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 4 FOR

```
%prueba04.m
for k=1:100,
    x=sqrt(k);
    if x>5,
        fprintf('x= %5.2f , k= %3d \n',x,k)% muestra por pantalla x
    y k
        break
    end
end
```

% contador
% obtiene la raíz de k
% si raíz es mayor a 5
% sale del lazo
% fin del if
% fin del for

Ejemplo 5 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 6

Sabiendo que las coordenadas cartesianas de una circunferencia son de la forma $x=r \cos(\theta)$, $y=r \sin(\theta)$ crea 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')

```

4.2 Propagación de Errores y Aritmética del Punto Flotante

Ejemplo 7

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 $T = 600^\circ \pm 20^\circ$.

(problema 4.10 del Chapra & Canale en pag. 103)

Solución

$$H = Ae\sigma T^4$$

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

Error aproximado:

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

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

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

Error exacto:

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

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

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

Este es muy cercana al resultado aproximado.

Ejemplo 8

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 $500 \pm 1\%$, 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:

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

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

- Aproxime HB .
- Estime el error absoluto de la aproximación HB
- Estime el rango para el valor exacto de la dureza HB .

Solución

$$D = 10$$

$$P = 500 \quad \xi_P = 5 \quad d = 5.75 \quad \xi_d = 0.001 \quad \pi = 3.14 \quad \xi_\pi = 0.5 \times 10^{-2}$$

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

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

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

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

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

$$HB = 17.5132 \pm 0.2098$$

$$17.3034 \leq HB \leq 17.7230$$

Ejemplo 9

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 epsilon de la máquina?

Solución:

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

Solución

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

5. 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:

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

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. Explique el siguiente resultado de Matlab (Matlab usa IEEE doble precisión)

```
>> (1 + 1e-16) - 1
.....
>> (1 + 2e-16) - 1
.....
>> (1 - 1e-16) - 1
.....
>> 1 + (1e-16 - 1)
.....
```

3. Explique los siguientes resultados ($\log(1+x)/x \approx 1$ para pequeños x)

```
>> log(1+3e-16)/3e-16
.....
>> log(1+3e-16)/((1+3e-16)-1)
.....
```

4. Evaluar $\sum_{k=1}^{3000} k^{-2} = 1.6446$, redondeando todos los resultados intermedios a 4

dígitos.

Nota (Utilizar la función chop para el redondeo, ver help del Matlab y lazo for ..end)

```
.....
.....
.....
.....
.....
.....
```

Este resultado tiene solamente dos dígitos correctos, pero no hay cancelación (existe sustracción) . Explique un mejor método.

Solución En una suma grande los términos gradualmente desminuyen, el error puede ser evitado sumando los términos mas pequeños juntos.

Simplemente invertir la orden de la suma restaura la exactitud.

.....

5. Asuma $S_N = \sum_{j=2}^N \frac{1}{j^2 - 1}$, y la solución exacta es $\frac{1}{2} \left(\frac{3}{2} - \frac{1}{N} - \frac{1}{N+1} \right)$.

(1) Elabore un programa (prog1.m) en Matlab para calcular $S1_N$ con la secuencia

$$S1_N = \frac{1}{2^2 - 1} + \frac{1}{3^2 - 1} + \dots + \frac{1}{N^2 - 1}$$

(2) Elabore un programa (prog2.m) en Matlab para calcular $S2_N$ con la secuencia

$$S2_N = \frac{1}{N^2 - 1} + \frac{1}{(N-1)^2 - 1} + \dots + \frac{1}{2^2 - 1}$$

(3) Use el programa1 y el programa2 para calcular S_{10^2} , S_{10^4} , S_{10^6} , respectivamente, y compare los resultados con la solución exacta.

	$S1_N$	$S2_N$	S
S_{10^2}			
S_{10^4}			
S_{10^6}			

6. Crear la función Horner que evalúa el valor del polinomio

$$p(x) = p_1x^n + p_2x^{n-1} + \dots + p_{n-1}x + p_n = (\dots ((p_1x + p_2)x + p_3)x + \dots + p_{n-1})x + p_n$$

En el punto x mediante el algoritmo de Horner (también conocido analíticamente como regla de Ruffini). Los coeficientes del polinomio se almacenan en un vector: $p=[p_1,p_2,\dots,p_{n-1},p_n]$. El algoritmo viene dado a continuación

Entrada: vector $p=[p_1,p_2,\dots,p_{n-1},p_n]$ y el punto x que se evalúa

Salida: valor de $p(x)$

Paso 1: Asignar $q = p_1$

Paso 2: Para k desde 2 hasta n repetir Paso 3

Paso 3: $q = (q*x) + p_k$

Con estos datos, la primera fila del archivo debe ser:

```
function q = horner(p,x)
```

.....

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

7. Igual que el ejemplo 6, en lugar de que la **función** tenga varios valores de (x,y); solo tenga un valor y vaya reemplazando dichos valores. La función se llamara circunferencia 2.m.

8. Crea un **script** que llame repetidas veces a el programa circunferencia1.m de forma que represente en una misma grafica 4 circunferencias distintas.

9. Crea una **función** que dibuje un cilindro y que se llame cilindro.m. La función tendrá como parámetros de entrada el radio, el alto y el ángulo

10. Crea una **función** que represente el tiro parabólico en tres dimensiones, sabiendo que las coordenadas vienen dadas por las ecuaciones: $x=V_0 \cos(\theta) \cos(\varphi) t$; $y= V_0 \cos(\theta) \sin(\varphi) t$; $z= V_0 \sin(\theta) t - (0.5 g t^2)$; siendo θ el ángulo inicial que forma con la vertical y φ el ángulo inicial que forma con el eje X.

	Curso	Cálculo Numérico	Código : MB535
	Tema	Métodos Directos SEL	
	Practica	03	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

1. Objetivo :

Aplicar los métodos directos de factorización y eliminación en la solución de sistemas lineales.

2. Fundamento Teórico

Muchas matrices especiales son funciones predefinidas; entre ellas están

3. Instrucciones básicas en Matlab

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 de la k_ésima diagonal
triu(A,k)	Extrae la parte triangular superior de A sobre la k_ésima diagonal

Resolución de sistemas

solve('ecuación', 'x') solve('ex1,ex2,...,exn', 'x1,x2,...,xn') X=linsolve(A,B)	Resuelve la ecuación en la variable x Resuelve n ecuaciones simultáneas ec1,...,ecn en las variables x1,...,xn (sistema de ecuaciones) Resuelve $A \cdot X = B$ 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 $A \cdot X = B$
X=A/B	Resuelve el sistema $X \cdot A = B$

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 0 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

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

4. Parte práctica

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

Instrucción	Solución
Hallar $\ A\ _{\infty}$ mentalmente	34
Hallar $\ A\ _{\infty}$ con un comando de Matlab	norm(A,inf)
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

<p>Comando: <code>[L,U]=lu(A)</code></p> <ul style="list-style-type: none"> _ L es una matriz triangular inferior _ U es una matriz triangular superior _ $L*U = A$. 	<pre>>> 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</pre>
---	---

Factorización LU con pivoteo en Matlab

<p>Comando: <code>[L,U,P]=lu(A)</code></p> <ul style="list-style-type: none"> _ L es una matriz triangular inferior _ U es una matriz triangular superior _ P es una matriz de permutación _ $L*U = P*A$. 	<pre>>> 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 ans = 7 8 0 1 2 3 4 5 6</pre>
---	--

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)`

que tenga como variables A y k 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=.....;
```

2. 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

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

son definidas positivas.

```
function [y]=menores1(A)
y=[]; % inicializa el vector y como nulo
[n,m]=size(A); % obtener dimensión de matriz n fil. y m col.
for k=1:n % Para k=1 hasta n
mm=A(1:k,1:k); % obtener los menores de la submatriz k
% mostrar por pantalla los valores de los menores
me= ; % calcular el determinante de los menores
y=[ ]; % encestar el vector y con el valor actual de me
end; % fin del para.
```

3. 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
$$x_k = \frac{b_k - \sum_{j=1}^{k-1} L_{k,j} x_j}{L_{k,k}}$$

```
function [ x ]= sustidir( 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)

4. 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
$$x_k = \frac{b_k - \sum_{j=k+1}^n U_{k,j} x_j}{U_{k,k}}$$

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

```
function [x]= gauss (A,b)
%Fuente:Gonzalo Hernández - Gonzalo Rios -MA-33A 2007-1-UCHile - Departamento de
%Ingeniería Matemática
%Los comandos que ejecuta la función "gauss.m" son los siguientes:
n = length(b); % Se guarda en la variable n el tamaño del vector b
for k = 1:(n-1) % El ciclo for comienza en k=1, en cada iteración se suma 1 a k, y
%termina cuando k=n-1, %incluyendo esa iteración
for i = k+1:n %Comienza un ciclo for anillado al anterior entre i=k+1 y i=n
m= A(i,k)/A(k,k);
%En la variable m se guarda el valor de la división sin modificar la matriz A
A(i,k+1:n) = A(i,k+1:n) - m*A(k,k+1:n);
%Para ahorrarnos un ciclo, se toma de la fila i los elementos desde la columna k+1 hasta la n
b(i)= b(i) - m*b(k); %Se hace la transformación en el vector b
end % Finaliza el ciclo del for de variable i
end %Finaliza el ciclo del for de variable k
for k = n:-1:1 % Comienza el ciclo de la sustitución en inversa, inicializando la variable
% k=n, en cada iteración se le suma -1 a k, y termina cuando k=1
x(k) = (b(k) - A(k,k+1:n)*b(k+1:n))/A(k,k);
% De forma similar, se ahorra un ciclo multiplicando la .la A(k,k+1:n) por la
% columna b(k+1:n)
end %Finaliza el ciclo del for de variable k
end % Finaliza la función "gauss.m"
```

Modifica la rutina de gauss.m incluyendo en ella la subrutina de sustitución inversa

6. Incorporar a la subrutina anterior una estrategia de pivoteo parcial.

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

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

8. Crear la Subrutina llamada Cholesky que permita resolver el sistema lineal $\mathbf{Ax}=\mathbf{b}$, comprobando primero si A es una matriz simétrica., en caso contrario enviar mensaje de fracaso.

Entrada : Orden la Matriz "n" y elementos de la Matriz simetrica "A"

Salida : Elemento $l_{ij}, i \leq j \leq i ; 1 \leq i \leq n$ de "L"

Inicio

$$l_{11} = \sqrt{a_{11}}$$

Para $j = 2$ Hasta n Hacer

$$l_{j1} = a_{j1}/l_{11}$$

Fin_Para

Para $i = 2$ Hasta $n - 1$ Hacer

$$l_{ii} = [a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2]^{1/2}$$

Para $j = i + 1$ Hasta n Hacer

$$l_{ji} = \frac{1}{l_{ii}} [a_{ji} - \sum_{k=1}^{i-1} l_{jk} l_{ik}]$$

Fin_Para

Fin_Para

$$l_{nn} = [a_{nn} - \sum_{k=1}^{n-1} l_{nk}^2]^{1/2}$$

Fin

9. Obtener la factorización de Cholesky de la matriz

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

- Manualmente
- Utilizando la función **chol** de Matlab

10. Crear la Subrutina llamada Doolittle que permita resolver el sistema lineal $\mathbf{Ax}=\mathbf{b}$, 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:

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

Function [L,U]=doolittle(A)

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

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

- 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 $A \cdot A^{-1} - A^{-1} \cdot A$?

12. Realizar la factorización LU de la matriz

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

- b. Sin Pivoteo Parcial
- c. Con Pivoteo Parcial

13. 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 $Ax=b$.

14. Sea la matriz A:

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

- a) A través de factorización LU determine a matriz inversa de A.
- b) Determine el número de condicionamiento de la matriz A. $cond(A) = \|A\| \|A^{-1}\|$
para $\|A\| = \max_i \sum_j l_{ij}$ y $0 < k \leq 1$.

	Curso	Cálculo Numérico	Código : MB535
	Tema	Métodos Iterativos para Resolver Sistemas de Ecuaciones Lineales y Calcular Valores y Vectores Propios de una matriz	
	Practica	04	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

1. Objetivos

Estudiar métodos para resolver sistemas de ecuaciones lineales mediante técnicas iterativas a partir de un vector solución inicial, el cual luego se va refinando hasta obtener soluciones de acuerdo a una tolerancia de precisión. 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: $Ax = b$, pueden expresarse como $x^{(k+1)} = Tx^{(k)} + c$. La matriz T y vector c varían de acuerdo al método. Además $A = D - L - U$.

Donde:

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

Método	T	c
Jacobi	$T_j = D^{-1}(L + U)$	$c_j = D^{-1}b$
Gauss-Seidel	$T_g = (D - L)^{-1}U$	$c_g = (D - L)^{-1}b$
SOR (Sobre-Relajación Sucesiva)	$T_\omega = (D - \omega L)^{-1}\{(1 - \omega)D + \omega U\}$	$c_\omega = (D - \omega L)^{-1}\omega b$

Criterios de convergencia de los Métodos Iterativos

Teorema (Condición necesaria y suficiente de convergencia).- La sucesión $x^{(k+1)} = Tx^{(k)} + c$, para $k \geq 0$, converge a la solución única, si y sólo si $\rho(T) \leq L < 1$, y el error de sucesión se puede estimar como sigue: $E \leq \frac{L^k}{1 - L} \|x^{(0)} - x^{(1)}\|_\infty$. Donde k representa las iteraciones, $k=0,1,\dots, \text{maxit}$.

Teorema (Condición suficiente de convergencia).- Si la matriz **A** es diagonal estrictamente dominante, los métodos de Jacobi y Gauss-Seidel convergen, para cualquier vector inicial $\mathbf{x}^{(0)}$ arbitrario.

En el método de Sobre-Relajación Sucesiva, cuando $\omega = 1$ obtenemos el método de Gauss-Seidel. La utilización de este parámetro permite obtener una convergencia más rápida.

Teorema (Kahan - Condición necesaria) .-Para que tenga convergencia el método SOR, cualquiera sea la estimación inicial, es necesario que ω esté en $]0,2[$.

Teorema (Ostrowski-Reich - Condición suficiente).- Si la matriz **A** es simétrica y definida positiva y $0 < \omega < 2$, entonces el método SOR converge para cualquier elección de la aproximación inicial $\mathbf{x}^{(0)}$ del vector solución.

Teorema: Si **A** es definida positiva y tridiagonal, entonces $\rho(\mathbf{T}_\omega) = [\rho(\mathbf{T}_j)]^2 < 1$, y la elección óptima de ω para el método SOR es:

$$\omega = \frac{2}{1 + \sqrt{1 - [\rho(\mathbf{T}_j)]^2}} \quad \text{y} \quad \rho(\mathbf{T}_\omega) = \omega - 1$$

Observación: En particular, concluimos que en el caso que **A** sea simétrica y definida positiva el método de Gauss-Seidel converge.

2. 2 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**.

Para poder aplicar el método debemos suponer que la matriz **A** de orden n con autovalores $\lambda_1, \lambda_2, \dots, \lambda_n$, con autovectores L.I: $\{v_1, v_2, \dots, v_n\}$ Además suponemos que **A** tiene un autovalor dominante, o también que $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n| \geq 0$

Sea x un vector no nulo en \mathbb{R}^n , se puede representar como una combinación lineal de los vectores propios $x = \sum_{i=1}^n \beta_i v^{(i)}$

Algoritmo de la Potencia

Entrada: $x^{(0)}$ vector inicial arbitrario no nulo, **A**, **tol**, **Maxit**

Salida: Valor propio dominante, y su respectivo vector propio

Paso 1: $k=1$

Paso 2: Normalizar ($\|\cdot\|_\infty$) el vector de $x^{(0)}$

Paso 3 Mientras que ($k \leq \text{Maxit}$) hacer los pasos 4-10

- Paso 4 $y = A * x$
 Paso 5 Encontrar el entero $p \in \langle 1, \dots, n \rangle$ tal que $|y_p| = \|y\|_\infty$
 Paso 6 Si $y_p = 0$, salir ('mensaje de fracaso')
 Paso 7 $\lambda = y_p$
 Paso 8 $err = \|x - y/y_p\|_\infty$
 $x = y/y_p$
 Paso 9 Si $err < \mathbf{tol}$ salida (λ, x)
 Paso 10: Tomar $k = k + 1$

3. Instrucciones básicas en Matlab

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

3.1 Implementación de la función para el método de Jacobi

Está dada por la siguiente forma iterativa:

$$X^{(k+1)} = D^{-1}(L + U)X^{(k)} + D^{-1}B \quad \text{donde } k = 0, 1, 2, \dots$$

Dado el punto inicial $X^{(0)}$, obtenemos los siguientes puntos $X^{(1)}, X^{(2)}, \dots$

La función *Jacobi* implementa el **Método Iterativo de Jacobi** para aproximar la solución de un sistema de ecuaciones:

```
% Jacobi.m
% Metodo de Jacobi
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;
```

Ejemplo.-

Resolver el siguiente sistema:

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

```
» 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)
```

3.2 Implementar el método de Gauss-Seidel y el SOR mediante funciones y resolver el sistema anterior.

3.3 Uso de la función “eig”.

Encontrar los valores y vectores propios de la matriz

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

```
»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 $\lambda = 2, 1, 3$ y las columnas de Q corresponde a lo vectores propios.

3.4 La función potencia implementa el método de la potencia:

```
% 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
```

Calcular por el método de la potencia, el mayor valor propio en valor absoluto de la

matriz $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 3 \\ 3 & 3 & 3 \end{bmatrix}$. Partiendo del vector $x_0 = (1 \ 1 \ 0)^T$

Solución

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].

3.5 Implemente una función para el método de la potencia inversa

a. Implemente una función para el método de la potencia inversa con desplazamiento (iterada)

4. Parte práctica

Ejemplo 1

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

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

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

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

$$\lambda = 0 \quad \lambda = -2/3 \quad \lambda = -1/9 \quad \rho(T_{GS}) = 2/3 < 1 \quad \therefore \text{Converge}$$

Ejemplo 2

Sea el sistema:

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

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

Solución

a.

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

Calculo del radio espectral

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

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

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

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

$$a > -1 + \sqrt{2} = 0.4142$$

b. $a=1$

Sist. Lineal $\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow$ Algoritmo de Jacobi $x^{(k+1)} = \begin{bmatrix} 0 & -1 \\ -1/3 & 0 \end{bmatrix} x^{(k)} +$

Iteraciones $x^{(0)} = [0 \ 0]^t$

$$\begin{bmatrix} 1 \\ 1/3 \end{bmatrix}$$

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

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

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

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

c. $L =$ Radio espectral $=$ máximo valor propio con $a=1$

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

$$L=0.5774$$

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

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. Crear una rutina para determinar si un sistema $\mathbf{A} \mathbf{x} = \mathbf{b}$, es convergente para el método de Sobre-Relajación Sucesiva (SOR), mediante el criterio del radio espectral:

```
function [flag, rho]=pruebaW(A,w)
% w : Factor de sobre-relajación
% flag : 1 si es convergente y 0 si no es convergente
% rho : Radio espectral
```

.....

.....

.....

.....

4. Elabore una rutina para resolver un sistema lineal $\mathbf{A} \mathbf{x} = \mathbf{b}$ mediante Sobre-Relajación Sucesiva (SOR), dado el factor de sobre-relajación, una tolerancia de error y un número máximo de iteraciones.

```
% function [x, numite]=solveSOR(A, b, w, TOL, MAXITE)
% w : Factor de Sobre-Relajación
```

.....

.....

.....

.....

.....

.....

5. 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 $x_1^{(0)} = 0$ $x_2^{(0)} = 0$:

$x_1^{(3)} = \dots\dots\dots x_2^{(3)} = \dots\dots\dots$

6. 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 $[1 \ 1 \ 1]^T$.

.....

.....

.....

.....

7. 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:

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

- a) $a > 9$ b) $-1 < a < 1$ c) $a > 8$ d) $-2 \leq a \leq 2$ e) $a > 2$

8. 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.

9. 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 dígito de su código

entonces el radio espectral de Jacobi será:

Las instrucciones en MATLAB serán:

.....

	Curso	Cálculo Numérico	Código : MB535
	Tema	Solución de Ecuaciones no Lineales de una y más variables	
	Practica	05	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

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

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. Métodos de Punto Fijo

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:

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

Algoritmo de aproximaciones sucesivas:

$$\text{Sea } f(x)=0 \text{ equivalente a } x = g(x)$$

Leer x_0

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

$$x_{n+1} = g(x_n)$$

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, 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

$$g(x) = \sqrt[4]{3+x-2x^2}. \text{Definiendo y representando la función } gx \text{ mediante}$$

```
» 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. Métodos de Newton

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.

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

Algoritmo de Newton-Raphson:

Leer x_0

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

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

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 $f_i : \mathbf{R}^n \rightarrow \mathbf{R}$, $1 \leq i \leq n$. funciones (no lineales) suficientemente diferenciables. Un sistema no lineal $n \times n$ se puede escribir de la forma:

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

Si definimos $F: \mathbf{R}^n \rightarrow \mathbf{R}^n$ por $F=(f_1, f_2, \dots, f_n)^t$ entonces podemos escribir (1) en forma vectorial como:

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

El método del punto fijo.-

Análogamente al caso unidimensional, 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 $G : \mathbf{R}^n \rightarrow \mathbf{R}^n$, o sea:

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

Donde g_1, g_2, \dots, 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

$$x^{(k)} = G(x^{(k-1)}), \quad k = 1, 2, \dots,$$

a partir de un punto inicial $x^{(0)}$. 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 $F(s)=0$.

El 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

$$F(x) = (f_1(x), f_2(x), \dots, f_n(x))^t$$

con $f_i : R^n \rightarrow R, \quad i = 1, 2, \dots, n$

Definimos la matriz Jacobiana de la función F como:

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

Sea x_0 una aproximación inicial al sistema $F(x)=0$. 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 x_1 como la solución de

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

es decir

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

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

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

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

3. Instrucciones básicas en Matlab

3.1 Funciones 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: Encuentra el cero de una función con una sola variable, es poco usada en la actualidad p.e. `fzero(dir_fun,rango_corchetes)`

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 $F(x)=0$

Ejm:

Se desea resolver:

$$2x_1 - x_2 = e^{-x_1}$$

$$-x_1 + 2x_2 = e^{-x_2}$$

Transformando el sistema

$$2x_1 - x_2 - e^{-x_1} = 0$$

$$-x_1 + 2x_2 - e^{-x_2} = 0$$

Empezamos con $x_0 = [-5 \ -5]$.

Primero, escribimos en un archivo m que calcule F en los valores x .

```
function F = myfun(x)
F = [2*x(1) - x(2) - exp(-x(1));
     -x(1) + 2*x(2) - exp(-x(2))];
```

Luego, llamamos a la rutina de optimización.

```
>>x0 = [-5; -5];           % condicion inicial
>> options=optimset('Display','iter'); % Opciones de salida
>>[x,fval] = fsolve(@myfun,x0,options) % Llamada al optimizador
```

Iteration	Func-count	f(x)	step	optimality	radius
0	3	23535.6	2.29e+004	1	
1	6	6001.72	1	5.75e+003	1
2	9	1573.51	1	1.47e+003	1
3	12	427.226	1	388	1
4	15	119.763	1	107	1
5	18	33.5206	1	30.8	
6	21	8.35208	1	9.05	1
7	24	1.21394	1	2.26	1
8	27	0.016329	0.759511	0.206	2.5
9	30	3.51575e-006	0.111927	0.00294	2.5
10	33	1.64763e-013	0.00169132	6.36e-007	2.5

Optimization terminated successfully:

First-order optimality is less than options.TolFun

x =

```
0.5671
0.5671
```

fval =

```
1.0e-006 *
-0.4059
```

-0.4059

3.2 Gráficas con Matlab

Gráficas 2D

Funciones de la forma $y = f(x)$

Dibujar la gráfica de la función $y = \sin(x)$

```
>>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

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

```
>>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:

$$\vec{r}(t) = (\sin(t), \cos(t), t); \quad 0 \leq t \leq 8\pi$$

```
>>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 $D = \{(x, y) / -2 \leq x \leq 2, -2 \leq y \leq 2\}$

Generamos el mallado

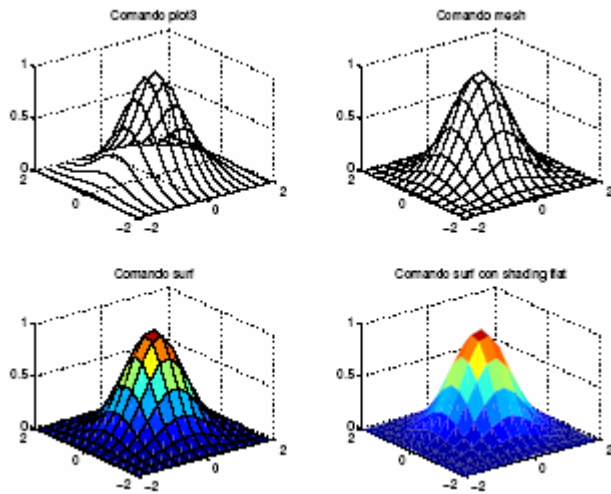
```
>>[x,y]=meshgrid(-2:.5:2);
```

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:

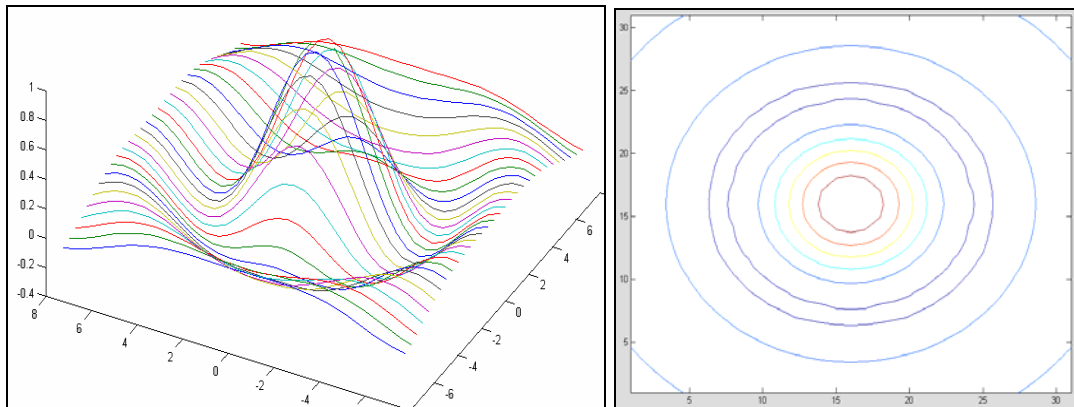
```
>>plot3(x,y,z)  
>>mesh(x,y,z)  
>>surf(x,y,z)  
>>surf(x,y,z)
```



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 0.500000000000000 1.000000000000000 0.500000000000000
0.500000000000000 0.750000000000000 1.000000000000000 0.250000000000000
0.500000000000000 0.625000000000000 0.750000000000000 0.125000000000000
0.500000000000000 0.562500000000000 0.625000000000000 0.062500000000000
```

```

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 $f(x) = \frac{e^{x/3}}{2} - \text{sen}(x)$ en el intervalo $[-10,10]$

Creemos 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

```

```

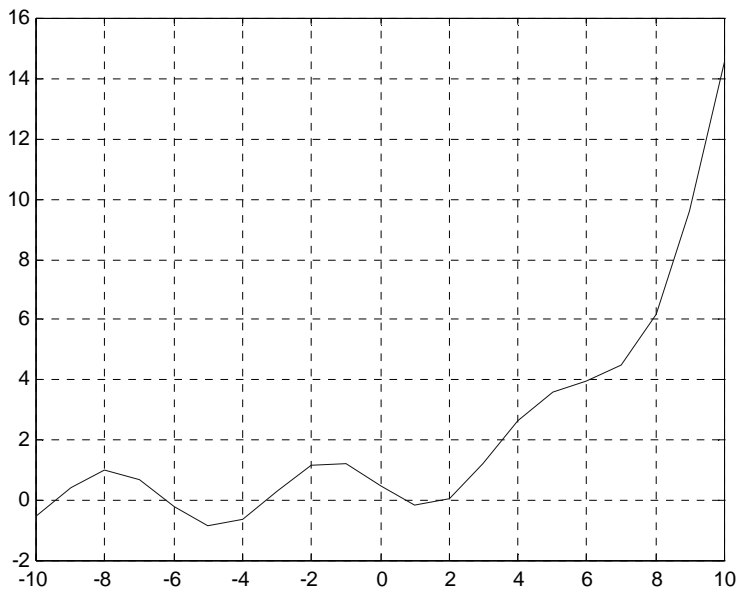
x vs y
-10.0000 -0.5262
-9.0000 0.4370

```

-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]=d fun1(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 almacenadas 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 Aproximaciones Sucesivas para Sistemas

Considere el sistema no lineal

$$3x_1 - \cos(x_2, x_3) - \frac{1}{2} = 0$$

$$x_1^2 - 81(x_2 + 0.1)^2 + \sin(x_3) + 1.06 = 0$$

$$e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} = 0$$

Aplique el método del punto fijo para aproximar la solución, realice 5 iteraciones y escoger $\mathbf{x}^{(0)} = (0.1, 0.1, -0.1)^t$

Solución

Si de la i -ésima ecuación se despeja x_i , el sistema puede cambiarse a un problema de punto fijo

$$x_1 = \frac{1}{3} \cos(x_2 x_3) + \frac{1}{6}$$

$$x_2 = \frac{1}{9} \sqrt{x_1^2 + \sin(x_3) + 1.06} - 0.1$$

$$x_3 = -\frac{1}{20} e^{-x_1 x_2} - \frac{10\pi - 3}{60}$$

Se obtiene la siguiente expresión de recurrencia

$$x_1^{(k)} = \frac{1}{3} \cos(x_2^{(k-1)} x_3^{(k-1)}) + \frac{1}{6}$$

$$x_2^{(k)} = \frac{1}{9} \sqrt{(x_1^{(k-1)})^2 + \sin(x_3^{(k-1)}) + 1.06} - 0.1$$

$$x_3^{(k)} = -\frac{1}{20} e^{-x_1^{(k-1)} x_2^{(k-1)}} - \frac{10\pi - 3}{60}$$

Partiendo de la estimación inicial, $x_1^{(0)} = 0.1$, $x_2^{(0)} = 0.1$, $x_3^{(0)} = -0.1$ se obtiene los siguientes resultados

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	0.100	0.100000	-0.100
1	0.4999	0.009441	-0.52310
8			
2	0.4999	0.000025	-0.52336
9			
3	0.5000	0.000012	-
0			0.5235981
4	0.5000	0.0000000	-
0		3	0.5235984
5	0.5000	0.0000000	-
0		2	0.5235987

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 :

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

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

Iteración 2 :

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

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

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 $x = (x, y)^T$ que

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

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

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

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 $f(x)=2x\cos(2x)-(x+1)^2$. 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 de 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 raíz 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 x_{niter}

Paso 3 Hacer $x_1=z$

Paso 4 Repetir para k desde 2 hasta niter los Pasos 5-6

Paso 5 Hacer $x_k=g(x_{k-1})$;

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 $x_{k+1}, x_{k+2}, \dots, x_{niter}$

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 $x^2 - \text{sen}(x) = 0$ 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 $x = 0, y = 0$

	Curso	Cálculo Numérico	Código : MB535
	Tema	Aproximación de Funciones	
	Practica	06	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

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 los datos (x_i, y_i) , $0 \leq i \leq n$, queremos hallar una función $g(x)$ tal que

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

Problema de Interpolación Polinomial: Dados los datos (x_i, y_i) , $0 \leq i \leq n$, queremos hallar un polinomio $p_n(x)$ de grado a lo más n , tal que

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

Existencia y construcción de $p_n(x)$

Sea el polinomio de la forma:

$$p_n(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{n-1} x + a_n,$$

Por condición de interpolación:

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

Esto es equivalente al sistema:

$$\begin{pmatrix} x_0^n & x_0^{n-1} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_n^n & x_n^{n-1} & \cdots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

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, \dots, x_n distintos.

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”:

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

Polinomio de interpolación de Newton basado en diferencias Divididas

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

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

Polinomios de interpolación de Lagrange

Para intervalos iguales o no.

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

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

Ajuste por mínimos cuadrados lineal

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

Se pueden ajustar a una recta $y = ax + b$ en forma óptima, resolviendo la ecuación normal:

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \Rightarrow M p = y$$

$$M^T M p = M^T y$$

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

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 función 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: $(x_0, y_0); (x_1, y_1); (x_2, y_2); \dots; (x_n, y_n)$;

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

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

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

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

Debemos primero resolver el siguiente sistema tridiagonal:

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

Una vez obtenidos M_1, \dots, 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:

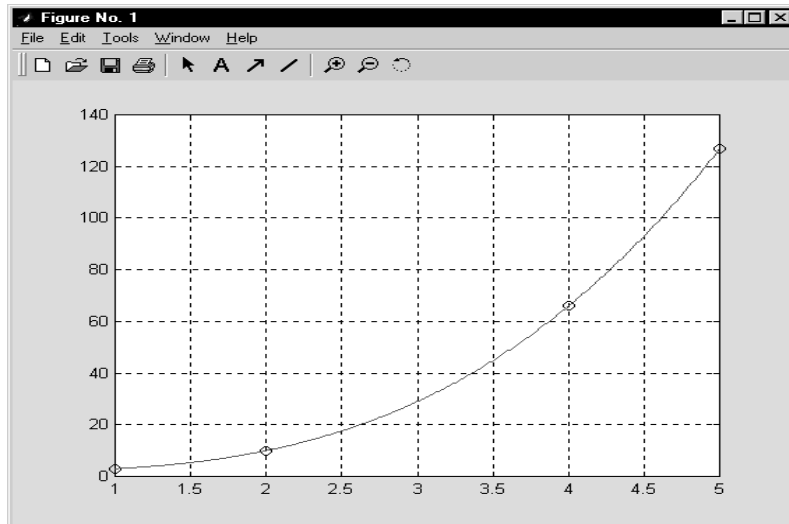
$$P(x) = x^5 + 2x^3 + x^2 - x + 1$$

» **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 $f(x) = e^{-x} \sin(x)$ 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 $P_4(x)$

17. Interpolare $\sin(x)$ 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:

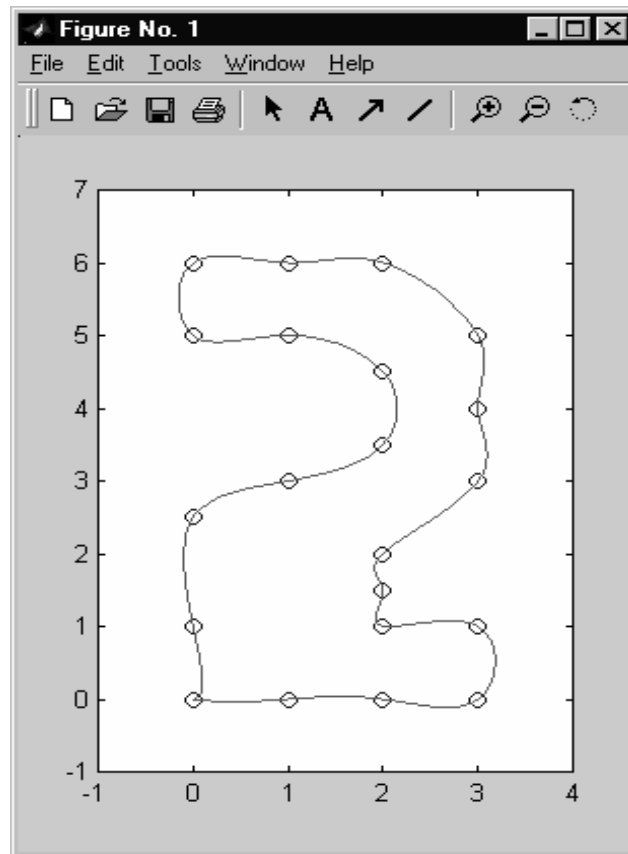
```
» x=[0 0.25 0.5 0.75 1]'
» y=[0.9162 0.8109 0.6931 0.5596 0.4055]'
» xi=[0.2 0.4 0.6]'
» yi=interp1(x,y,xi,'linear')
» plot(x,y,'o',xi,yi,'x')
```

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":

- » $x = [0.1 \ 0.4 \ 0.5 \ 0.7 \ 0.7 \ 0.9]^t$
- » $y = [0.61 \ 0.92 \ 0.99 \ 1.52 \ 1.47 \ 2.03]^t$
- » $M = [x \ \text{ones}(6,1)]$
- » $A = M^t * M$

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

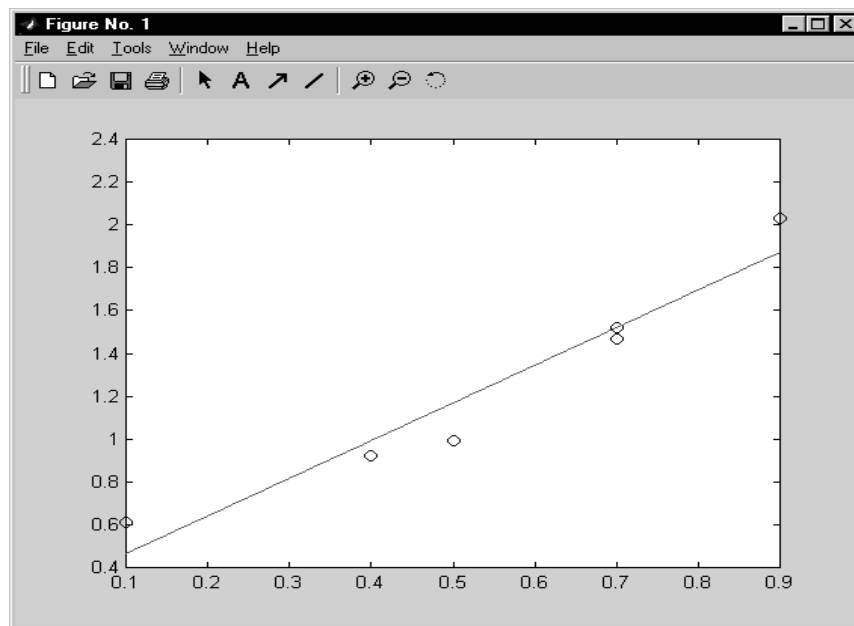
También se puede obtener directamente:

» 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:

- Un polinomio de grado 5 que pase por todos los puntos
- Una ajuste lineal
- Un ajuste cuadrático
- Una función spline
- La función exacta

4. Parte práctica

Problema 1

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

- Muestre las funciones Spline $S(x)$ para cada intervalo.
- Dime el máximo error cometido, para ello tabule $|p(x) - S(x)|$ con $\Delta x = 1/4$.
- 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 2

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	-

Se obtiene:

$$B = -2.8174 \quad A = -2.2349$$

$$Y = -2.8174 T - 2.2349$$

De donde se obtiene:

$$b = -B = 2.8174$$

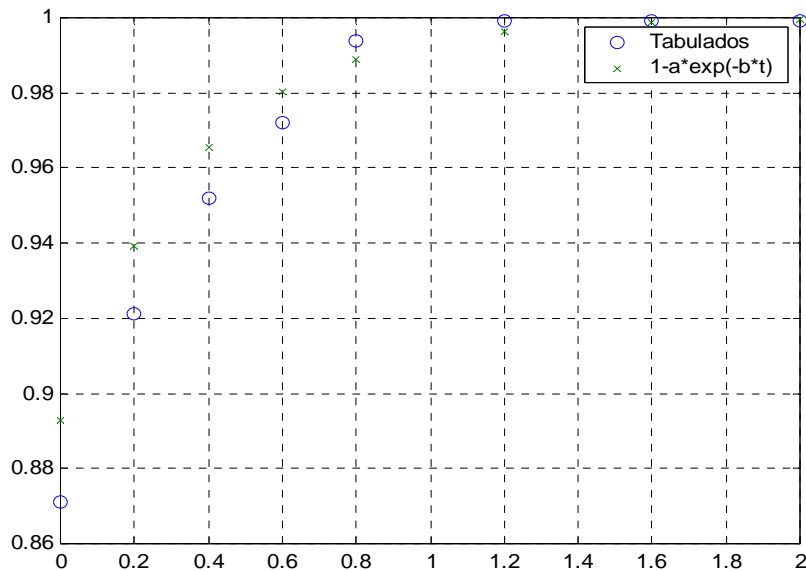
$$a = e^A = 0.1070$$

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
y	0.871	0.921	0.952	0.972	0.994	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

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:

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

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

$$t \approx 0.27 \text{ seg.}$$

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			
1	3	2		
2	5	f	h	
d	11	g	2	c

El producto de c y d es:

3. En la interpolación de una función f que pasa por los puntos $x_i = 2i$, $i = 0,1,2$.
 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) $L=\text{poly}([2\ 0])/\text{polyval}(\text{poly}([0\ 2],1))$
 b) $L=\text{poly}([2\ 0])/\text{poly}([0\ 2],1)$
 c) $L=\text{polyval}([0\ 2],1)/\text{poly}([0\ 1\ 2],1)$
 d) $L=\text{poly}([0\ 2])/\text{polyval}(\text{poly}([0\ 2],1))$
 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.

.....

	Curso	Cálculo Numérico	Código : MB535
	Tema	Diferenciación e Integración Numérica	
	Practica	07	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

1. Objetivos

Estudiar y aplicar los métodos para obtener derivadas e integrales a partir de una tabla de valores de la función, parecidos a los que se usa en la interpolación, también dichos métodos son de utilidad para estimar el valor de la derivada o la integral cuando se conoce la función $f(x)$.

2. Fundamento Teórico

Fórmulas para la primera derivada:

Hacia delante

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

Central

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

Fórmulas para la primera derivada:

Hacia delante

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

Central

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

Integración Numérica

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

Regla de los trapecios (n = 1)

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

Regla de Simpson (n = 2)

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

Regla de Simpson Simpson tres-octavos (n = 3)

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

Regla de Milne (n = 4)

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

Regla de Cuadratura Gaussiana

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

3. Instrucciones básicas en Matlab

diff(X): Para un vector X, es [X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)].

quad : Evalúa la integral, por el método de Simpson.
quad8 : Evalúa la integral, por el método de Newton-Cotes.

Cálculo de Primitivas

El comando encargado del cálculo de primitivas en Matlab es **int**. Por ejemplo **int**

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

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)
```

4. Parte práctica

- Hallar la derivada de la función: x^2 en $x=3$. Usar $h=0.05$
 - » `x=[3 3+0.05]` % Se crea un arreglo de dos dimensiones
 - » `y=x.^2` % Se crea otro arreglo, utilizando operaciones 'punto'
 - » `dydx=diff(y)./diff(x)`.
- Ajustar un polinomio P de cuarto grado a cinco puntos uniformemente espaciados para $f(x)=\sin(\pi x)$ entre 0 y 1; luego, aproximar $P'(x)$ para $x=0.63$. Compare los resultados con los valores analíticos.
 - » `x=[0 .25 .5 .75 1]`;
 - » `y=sin(pi*x)`;
 - » `p=polyfit(x,y,4)`;
 - » `polyval(polyder(p),0.63)`
 - %El valor aproximado de $f'(0.63)$:
 - » `dydx=diff(y)./diff(x)`;

» dydx(3) %dado que 0.63 está en 3er intervalo de x.
 %Además el valor analítico de f'(0.63):
 » cos(pi*0.63)*pi

3. Elabore funciones que implemente el método del trapecio, el método de Simpson, el método de Simpson 3/8, el método de Milne.

Solución:

La función *trapecio* implementa el método del trapecio

```
function y = trapecio(f,a,b)
y=(1/2)*(b-a)*[feval(f,a)+feval(f,b)];
```

La función *simpson* implementa el método de Simpson

```
function y = simpson(f,a,b)
y=(1/6)*(b-a)*[feval(f,a)+4*feval(f,(a+b)/2)+feval(f,b)];
```

La función *simpson3_8* implementa el método de Simpson 3/8

```
function y = simpson3_8(f,a,b)
y=(1/8)*(b-a)*[feval(f,a)+3*feval(f,(2*a+b)/3)+ ...
3*feval(f,(a+2*b)/3)+feval(f,b)];
```

La función *milne* implementa el método de Milne

```
function y = milne(f,a,b)
y=(1/90)*(b-a)*[7*feval(f,a)+32*feval(f,(3*a+b)/4)+ ...
12*feval(f,(a+b)/2)+32*feval(f,(a+3*b)/4)+7*feval(f,b)];
```

4. Calcule la integral de manera aproximada empleando para ello la regla del trapecio compuesta.

$$\int_0^1 (\tan x + 2x^5) dx$$

Solución:

Definimos la función, los límites de integración y el paso h:

```
>> f=inline('tan(x)+2*x.^5'), a=0, b=1, n=4
>> h=(b-a)/n;
>> X=[a:h:b];
>> Y=f(X)
>> P=h/2*(f(a)+2*sum(Y(2:length(X)-1))+f(b))
P:= 1.012751808
```

5. Calcule la siguiente integral por el método de Simpson compuesto.

$$\int_0^4 e^x dx$$

Solución:

Para resolver la integral por este método, primeramente introducimos un número para $n = 2m$ de intervalos y definimos el paso $h = (b-a)/2m$:

```

>>m =4, a =0, b =4
>>h=(b-a) / (2*m)
>>f= inline('exp(x) ')
>>X=[a:h:b]
>>Y=f(X)
>>P=h/3*( f(a)+2*sum(Y(3:2:2*m-1))+4*sum(Y(2*(1:m)))+f(b))
P:= 53.61622078

```

6. Elabore un programa que calcule la integral de una función utilizando el método del Romberg.

Integración de Romberg

```

%romb.m
%Intregales mediante el metodo de Romberg

function y=romberg(f,a,b,n)
format long
h=(b-a);

R(1,1)=h*(feval(f,a)+feval(f,b))/2;
for i=2:n
    R(i,1)=(1/2)*(R(i-1,1)+h*sum(feval(f,a+([1:2^(i-2)]-0.5)*h)));
    for j=2:i
        R(i,j)=(4^(j-1)*R(i,j-1)-R(i-1,j-1))/(4^(j-1)-1);
    end
    h=h/2;
end
y=R;

```

7. Elabore un programa que calcule la integral de una función utilizando el método de Cuadratura Gaussiana

```

function z=CuadraturaGauss(f,a,b,nn)
%f la función
%a,b intervalos de integración
%nn número de puntos
switch nn
case 1
    x = 0;    w = 2;

case 2
    x(1) = -0.57735026918963;    x(2) = -x(1);
    w(1) = 1;                    w(2) = w(1);

```

```

case 3
    x(1) = -0.77459666924148;    x(2) = 0;
    x(3) = -x(1);
    w(1) = 0.555555555555556;    w(2) = 0.888888888888889;
    w(3) = w(1);

case 4
    x(1) = -0.861136311594053;    x(4) = -x(1);
    x(2) = -0.339981043584856;    x(3) = -x(2);
    w(1) = 0.347854845137454;    w(4) = w(1);
    w(2) = 0.652145154862546;    w(3) = w(2);

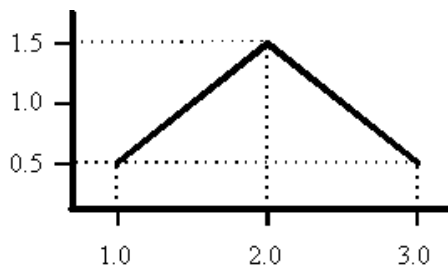
otherwise
    error(sprintf('el nro de puntos debe ser menor que 8'));

end
s=0;
for i=1:nn
    t(i) = (a+b+x(i)*(b-a))/2;
    s=s+[w(i)*feval(f,t(i))]*(b-a)/2;
end
z=s;

```

5. 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:

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

.....

.....

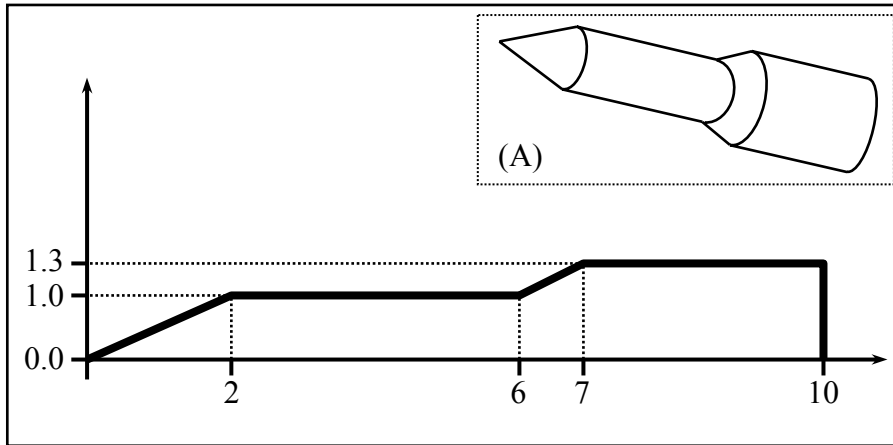
3. ¿Cuántos intervalos necesitamos para aproximar la integral $\int_1^3 \ln(x) dx$ por el método de Simpson con un error $\varepsilon < 0.001$

.....

.....

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	Cálculo Numérico	Código : MB535
	Tema	Ecuaciones Diferenciales Ordinarias	
	Practica	08	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

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

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

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

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

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

2.1 Método de Euler

Progresivo:

$$y_{i+1} = y_i + hf(t_i, y_i), \quad i = 0, 1, \dots, N - 1$$

Regresivo:

$$y_{i+1} = u_i + hf(t_{i+1}, y_{i+1}), \quad i = 0, 1, \dots, N - 1$$

2.2 Método de Taylor de orden 2

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

2.3 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

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

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

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

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

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

Problema del Valor Frontera

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

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

$$u(t_0) = u_0$$

$$u(b) = B$$

Método del Disparo

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

$$u(t_0) = u_0$$

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

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	$y_N = y_N(s_0)$
0.2000	0.1864	0.5088	
0.3000	0.2420	0.6071	
0.4000	0.3086	0.7285	
0.5000	0.3887	0.8780	

Se obtiene $y_N = y_N(s_0)$ y comparamos con $y(0.5) = 0.283$.

Se elije 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	$y_N = y_N(s_1)$
0.2000	0.1390	0.2409	
0.3000	0.1659	0.2981	
0.4000	0.1990	0.3677	
0.5000	0.2399	0.4523	

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:

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

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	$y_N = y_N(s_3)$
0.2000	0.1527	0.3185	
0.3000	0.1879	0.3876	

La solución con diferencias finitas

t(k)	y(k)=X1
1.0000000000000000	1.0000000000000000
1.2000000000000000	1.18692106203224
1.4000000000000000	1.38127313638853
1.6000000000000000	1.58226407575024
1.8000000000000000	1.78883227403986
2.0000000000000000	2.0000000000000000

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

$$u' = 0.5 * u, \quad u(0) = 0.25$$

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 $x \in [0, 1]$

- Calcule las soluciones aproximadas usando los métodos de Euler progresivo y regresivo con paso $h = 0.1$. Determine un máximo error de truncación.
- Calcule la solución aproximada usando el método de Taylor de segundo orden con paso $h = 0.2$.

Solución

a) i.-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

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1})$$

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} (i)

$$y_{i+1} = \frac{y_i}{1 + h(1 - (x_i + h)^2)}$$

Tabla de Euler Progresivo

x_i	y_i	y_{i+1}
0.0	1.0000	0.9000
0.1	0.9000	0.8109
0.2	0.8109	0.7331
0.3	0.7331	0.6663
0.4	0.6663	0.6104
0.5	0.6104	0.5646
0.6	0.5646	0.5285
0.7	0.5285	0.5015
0.8	0.5015	0.4835
0.9	0.4835	0.4743
1.0	0.4743	—

Tabla de Euler Regresivo

x_i	y_i	y_{i+1}
0.0	1.0000	0.9099
0.1	0.9099	0.8302
0.2	0.8302	0.7610
0.3	0.7610	0.7050
0.4	0.7050	0.6530
0.5	0.6530	0.6137
0.6	0.6137	0.5840
0.7	0.5840	0.5637
0.8	0.5637	0.5532
0.9	0.5532	0.5532
1.0	0.5532	—

Cálculo del error máx. de truncación en los métodos de Euler (progresivo y regresivo)

$$\begin{aligned} \|\Gamma_h\| &\leq \frac{h}{2} \times \sup_{x \in [0,1]} \left| \frac{d}{dx} [y(x) \cdot (x^2 - 1)] \right| \\ &= 0.05 \times \sup_{x \in [0,1]} \left| y(x) \cdot [2x + (x^2 - 1)^2] \right| \\ &\leq 0.05 \times y(0) \times 1 \\ &= 0.1 \end{aligned}$$

b) Método de Taylor de segundo orden el valor de y_{i+1} es determinado por la expresión:

$$y_{i+1} = y_i + h \cdot f(x_i, y_i) + \frac{h^2}{2} f''(x_i, y_i)$$

Haciendo la sustitución para este ejemplo tenemos:

$$y_{i+1} = y_i + h \cdot y_i \cdot (x_i^2 - 1) + \frac{h^2}{2} y_i \cdot [2x_i + (x_i^2 - 1)^2]$$

x_i	y_i	y_{i+1}
0.0	1.0000	0.8200
0.2	0.8200	0.6842
0.4	0.6842	0.5899
0.6	0.5899	0.5334
0.8	0.5334	0.5134
1.0	0.5134	—

Aplicando sucesivamente esta fórmula se obtiene la siguiente tabla de valores:

```

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:

$$y'(t) = \frac{2y}{t} + t^2 e^t$$

$$y(1) = 0 \quad , t \in [1,2]$$

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:

$$\text{Paso } h = 0.25 \Rightarrow y(2) = y_4$$

$$y_e = y_i + hf(t_i, y_i)$$

$$t_{i+1} = t_i + h$$

$$y_{i+1} = y_i + \frac{h}{2} [f(t_i, y_i) + f(t_{i+1}, y_e)]$$

$$y_e = 0.67957 \quad t_1 = 1.25 \quad y_1 = 1.1574$$

$$y_e = 2.9838 \quad t_2 = 1.5 \quad y_2 = 3.8284$$

$$y_e = 7.6254 \quad t_3 = 1.75 \quad y_3 = 9.0192$$

$$y_e = 16.002 \quad t_4 = 2.0 \quad y_4 = 18.205$$

$$\varepsilon = 18.6831 - 18.205 = 0.4781 \quad \underline{\% \varepsilon = 2.6 \%}$$

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 :

$$y' = f(x, y) = 4e^{0.8x} - 0.5y$$

$$y(0) = 2 \quad x \in [0,3]$$

$$h = 1.5 \Rightarrow y(3) = y_2$$

$i = 0 \quad k_1 = 3, k_2 = 5.1635, k_3 = 4.3522, k_4 = 9.0163$

$$y_1 = 2 + \frac{1.5}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 9.7619$$

$i = 1 \quad k_1 = 8.3995 \quad k_2 = 16.168 \quad k_3 = 13.255 \quad k_4 = 29.271$

\vdots

$$y_2 = y_1 + \frac{1.5}{6}[k_1 + 2k_2 + 2k_3 + k_4] = 33.891$$

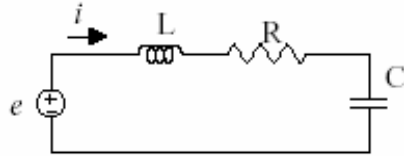
$$\% \mathcal{E} = \frac{|33.891 - 33.6772|}{33.6772} * 100 = 0.634\%$$

```
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
```

$\gg f=\text{inline}('4*\exp(0.8*x)-0.5*y','x','y')$

$\gg [y]=\text{rk4s}(f,0,3,2,1.5)$

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)

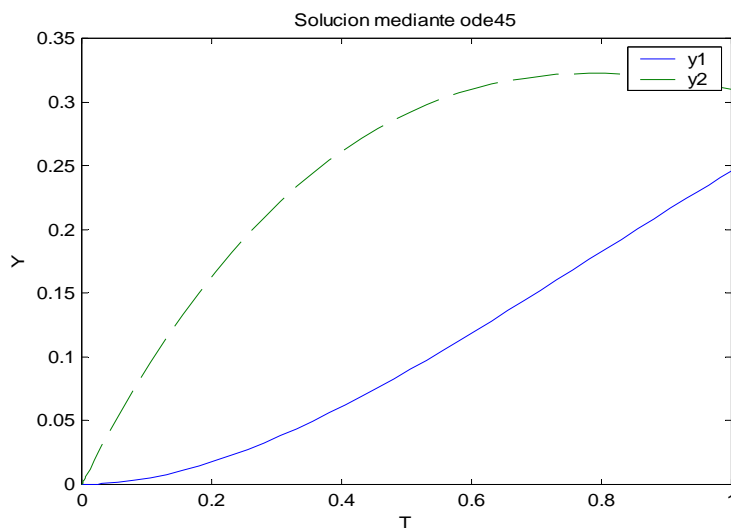
```
y =
0         0         0
0.1000    0.0047    0.0903
0.2000    0.0175    0.1627
0.3000    0.0367    0.2189
0.4000    0.0608    0.2610
0.5000    0.0885    0.2908
0.6000    0.1186    0.3099
0.7000    0.1501    0.3199
0.8000    0.1823    0.3223
0.9000    0.2144    0.3185
1.0000    0.2458    0.3096
```

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 =

$C1 \cdot \exp(t^2)$

```
» y=dsolve('Dy=2*t*y','y(1)=1')
```

y =

$1/\exp(1) \cdot \exp(t^2)$

- Ecuación de orden superior

```
» y=dsolve('D2y+2*Dy+2*y=1','y(0)=0','Dy(0)=0','x')
```

y =

$$1/2 - 1/2 \cdot \exp(-x) \cdot \sin(x) - 1/2 \cdot \exp(-x) \cdot \cos(x)$$

```
» 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. 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. : $a=1$ $y(a)=1$ $b=2$ $y(b)=1$

$$y' = z \qquad y(1) = 1$$

$$z' = y'' = \frac{0.5}{x} \sqrt{11 + (z)^2} = f(x, y, z) \quad z(1) = s_0 = 0$$

Algoritmo de Euler

x	y	y'
1	1	$s_0=0$
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?

$$d = \int_1^2 \sqrt{1 + (y'(x))^2} dx = \int_1^2 f(x) dx$$

$$h = 0.25$$

$$T = h * 0.5 * (f_1 + f_5 + 2 * \text{sum}(f(2:4))) = 1.0809$$

6. Considere el siguiente Problema de Valores de Contorno:

$$-u''(x) + u'(x) = x(1-x); \quad x \in (0,1)$$

$$u(0) = 0 \quad \text{y} \quad u(1) = 0$$

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:

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + \frac{u_{i+1} - u_i}{2h} = x_i(1-x_i)$$

$$\left(-1 - \frac{h}{2}\right)u_i - 1 + 2u_i + \left(-1 + \frac{h}{2}\right)u_{i+1} = h^2 x_i(1-x_i) \quad h=0.25$$

$$N+1 = (1-0)/h$$

$$N+1 = 4 \quad N = 3$$

$$i = 1, 2, 3$$

$$(-1 - h/2) = -1.125$$

$$(-1 + h/2) = -0.875$$

$$\begin{bmatrix} 2 & -0.875 & 0 \\ -1.125 & 2 & -0.875 \\ 0 & -1.125 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} h^2(0.1875) \\ h^2(0.25) \\ h^2(0.1875) \end{bmatrix}$$

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

$$\begin{cases} y'(t) = \frac{1}{1+y^2} \\ y(0) = 1 \end{cases}$$

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: $D = \{(t, y) \in \mathbb{R}^2 / 1 \leq t \leq 2; -3 \leq y \leq 4\}$

.....

3. Considere la ecuación diferencial $y''+4ty'+2y^2 = 0$ con condiciones iniciales $y(0) = 1, y'(0) = 0$ 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: $y(0) = 1, y'(0) = 2, y''(0) = 0$, obtener $y(0.2)$ usando Euler con $h = 0.1$.

$$y(0.2) = \dots\dots\dots$$

5. Para la EDO : $2 \frac{dy}{dx} + 3y = e^{-5x}, y(0) = 7$, encuentre la constante L de Lipschitz del teorema de existencia y unicidad de la EDO.

$L = \dots\dots\dots$

6. Considere la ecuación diferencial $y''+4ty'+2y^2 = 0$ con condiciones iniciales $y(0) = 1, y'(0) = 0$ 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 $h=0.1$

	Curso	Cálculo Numérico	Código : MB535
	Tema	Derivadas Parciales	
	Practica	09	
	Profesores	Castro Salguero, Robert Garrido Juárez, Rosa Pantoja Carhuavilca, Hermes	

1. Objetivos

Estudiar y aplicar los métodos para resolver numéricamente ecuaciones en derivadas parciales.

2. Fundamento Teórico

Una ecuación diferencial parcial (EDP) puede ser escrita en forma general

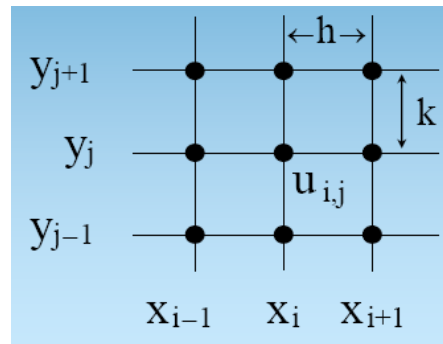
$$a \frac{\partial^2 \phi}{\partial x^2} + b \frac{\partial^2 \phi}{\partial x \partial y} + c \frac{\partial^2 \phi}{\partial y^2} + d \frac{\partial \phi}{\partial x} + e \frac{\partial \phi}{\partial y} + f \phi + g = 0$$

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 \mathcal{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 \rightarrow EDF
- Métodos explícitos
 - Sencillos
 - Inestables
- Métodos implícitos
 - Más complejos
 - Estables



2.1 EDPs Elípticas

Podemos citar a la *ecuación de Poisson*

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = g(x, y) \quad (1)$$

o de Laplace

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

2.1.1 Método Explícito

Para la ecuación (1) aproximaremos la segunda derivada a través de la fórmula 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 espaciamentos 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 Explícito

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 $(i + 1)$ -ésimo tiempo. Note que los puntos discretos son $x_j = jh$ y $t_i = ik$

2.3 EDPs Hiperbólicas

La ecuación a tratar en esta oportunidad es la ecuación de la onda unidimensional, cuya expresión es:

$$\frac{\partial^2 u}{\partial x^2} = c \frac{\partial^2 u}{\partial t^2} \quad 0 < x < L, \quad t > 0$$

Condiciones Iniciales

$$u(x, 0) = f(x) \quad u_t(x, 0) = g(x)$$

Condiciones de Contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t)$$

2.3.1 Método Explícito

$$\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = c^2 \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

Condiciones iniciales

$$u_{i,0} = f_i \quad y \quad u_{i,1} - u_{i,-1} = 2kg_i$$

Paso 1°

$$u_{i,1} = \alpha^2(f_{i-1} + f_{i+1})/2 + (1 - \alpha^2)f_i + kg_i$$

Pasos siguientes

$$u_{i,j+1} = \alpha^2 (u_{i+1,j} + u_{i-1,j}) + 2(1 - \alpha^2)u_{i,j} - u_{i,j-1}$$

$$\text{Convergencia} \quad \alpha < 1$$

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

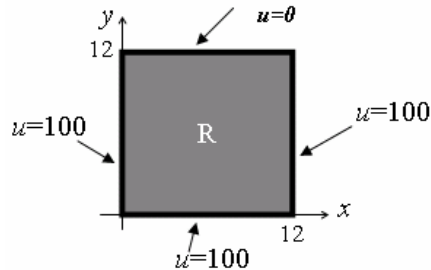
12.5000 25.0000 50.0000 75.0000 75.0000

```

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 $h=4\text{cm}$.

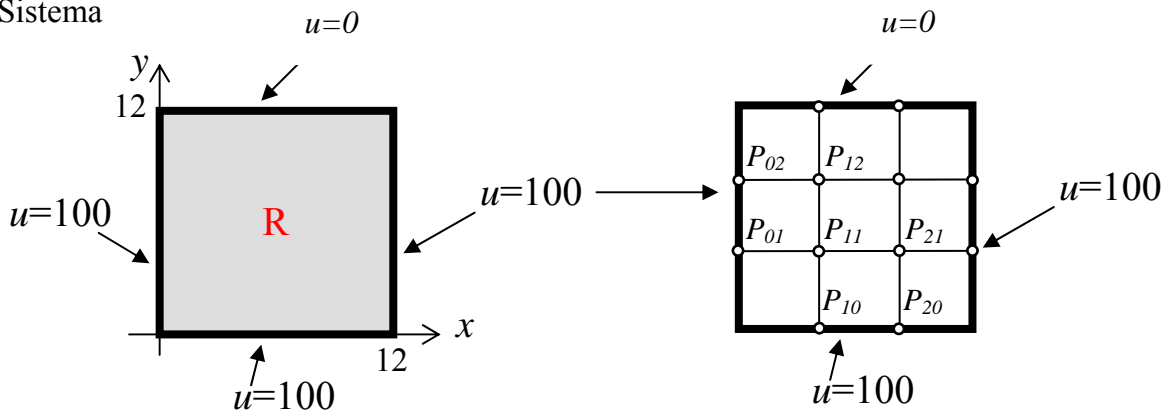


La ecuación de transferencia de calor en estado estacionario se reduce a Laplace

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0. \text{ Plantee el sistema lineal en los nodos pedidos.}$$

Solución

Sistema



<p>Nodo P_{11} $100 + P_{12} + P_{21} + 100 - 4 P_{11} = 0$</p> <p>Nodo P_{21} $P_{11} + P_{13} + 100 + 100 - 4 P_{21} = 0$</p> <p>Nodo P_{12} $100 + 0 + P_{13} + P_{11} - 4 P_{12} = 0$</p> <p>Nodo P_{13} $0 + P_{12} + 100 + P_{21} - 4 P_{13} = 0$</p>	$\begin{bmatrix} -4 & 1 & 1 & 0 \\ 1 & -4 & 0 & 1 \\ 1 & 0 & -4 & 1 \\ 0 & 1 & 1 & -4 \end{bmatrix} \begin{bmatrix} P_{11} \\ P_{21} \\ P_{12} \\ P_{13} \end{bmatrix} = \begin{bmatrix} -200 \\ -200 \\ -100 \\ -100 \end{bmatrix}$
---	--

Ejemplo 2:

Resolver el siguiente EDP

$$u_{tt} = c^2 u_{xx}, \quad 0 < x < L, \quad t > 0$$

$$c = 1, \quad L = T = 4, \quad n_x = 4, \quad n_t = 8,$$

$$u(x, 0) = 2 - |x - 2| \quad u_t(x, 0) = 0$$

$$u(0, t) = 0 \quad u(L, t) = 0$$

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 $a_2 = 1/4$, $1 - a_2 = 3/4$:

$$u_{1,1} = (1/4)(u_{0,0} + u_{2,0})/2 + (3/4)u_{1,0} = \\ (1/4)(0 + 2)/2 + (3/4)1 = 1 = u_{3,1}$$

$$u_{2,1} = (1/4)(u_{1,0} + u_{3,0})/2 + (3/4)u_{2,0} = \\ (1/4)(1 + 1)/2 + (3/4)2 = 7/4$$

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:

$$u_{1,2} = (1/4)(u_{0,1} + u_{2,1}) + (3/2)u_{1,1} - u_{1,0} \\ = (1/4)(0 + 7/4) + (3/2)1 - 1 = 15/16 \\ = u_{3,2}$$

$$u_{2,2} = (1/4)(u_{1,1} + u_{3,1}) + (3/2)u_{2,1} - u_{2,0} \\ = (1/4)(1 + 1) + (3/2)(7/4) - 2 = 9/8$$

5. Ejercicios Propuestos

1. Dada la E.D.P

$$(x+1) \frac{\partial^2 w}{\partial x^2} + (y^2 + 1) \frac{\partial^2 w}{\partial y^2} - w = 1$$

$$0 \leq x \leq 1, \quad 0 \leq y \leq 1, \quad \Delta x = \Delta y = 1/3$$

Con las condiciones de frontera:

$$w(0, y) = y$$

$$w(1, y) = y^2$$

$$w(x, 0) = 0$$

$$w(x, 1) = 1$$

- Dibuje la malla con las incógnitas y los valores frontera
- Plantear el sistema de ecuaciones mediante diferencias finitas

.....

.....

.....

.....

.....

