Introducción al Cálculo Numérico y Programación

Guión de Prácticas
Curso 2005-2006

Normas de la asignatura

- ***Las prácticas son obligatorias para todos los alumnos. Para poder aprobar la asignatura deberán, como mínimo, hacerlas en el aula de informática y presentar los guiones de los módulos cuando se les requiera.***
- ***Se realizará una evaluación continua de las prácticas y un examen final de éstas en Mayo.***
- ***La calificación final de la asignatura vendrá dada por la unión de las notas de la parte práctica y de parte teórica.***

***Requisitos*:**

- **Un disquete (propiedad del alumno) que se deberá llevar siempre a las clases de prácticas.**
- **Una ficha con una foto del alumno.**

Consideraciones:

1. **El acceso a MATLAB se realiza presionando dos veces en el botón izquierdo del ratón en el icono Matlab**
2. **MATLAB diferencia entre mayúsculas y minúsculas para los nombre de las variables (normalmente no es así para los comandos o funciones intrínsecas)**
3. **Al terminar cada sesión se debe comprobar que se graba el trabajo realizado en esa sesión en el disquete personal, disquete A, tras confirmar que no hay virus en el disco duro (unidad C) y en el disquete A.**
4. **Para garantizar que se va a trabajar en la unidad A escribir, al principio de cada sesión, en la línea de comandos:**

   >> cd a:

   Con ello garantizamos que trabajamos con nuestro disco.
5. **Una vez que se ha realizado un módulo o parte de él, se debe entregar al profesor el INFORME escrito con los enunciados y los resultados obtenidos.**
6. **En los informes de los módulos en donde se requiera hacer programas o figuras se deben escribir los programas y las figuras.**

MÓDULO 1: ÓRDENES EN MATLAB

Alumno: **.**

En la siguiente tabla se indican algunas órdenes útiles en MATLAB.

ORDEN	***SIGNIFICADO DE LA ORDEN***
HELP	• **Ayuda sobre órdenes y funciones de Matlab.** • **Ayuda sobre nuestras propias funciones .m**
WHAT	**Da una lista de funciones .m en el directorio especificado (*Por ejemplo: >> what a:\ejemplos*)**
CD	• **Cambia al directorio padre o directorio anterior *(>> cd ..)*** • **Cambia al directorio que se especifique *(>> cd a:\ejemplos*)**
DIR	**Da una lista del contenido del directorio en el que me encuentro**

1. Haciendo uso del comando HELP ejecuta y anota en la columna en blanco el significado de lo que se te indica al ejecutar las siguientes sentencias:

>> help norm	
>> help format	

2. Como hemos visto antes, en MATLAB existen diferentes modos o formatos de visualizar un resultado numérico. Dicho formato de salida se selecciona con el comando FORMAT. Para poder visualizar el resultado numérico que se le ha asignado a una variable "x" podemos ejecutar:

a) estas dos sentencias:

>> format long >> x=1.44444444444444444 >> x	**Indica abajo el resultado de salida**

a) Ejecuta estas dos sentencias:

>> format short >> x	**Indica abajo el resultado de salida**

b) Ejecuta estas dos sentencias:

>> format short e >> x	**Indica abajo el resultado de salida**

c) Ejecuta estas dos sentencias:

>> format long e >> x	**Indica abajo el resultado de salida**

d) Haciendo uso de la primera línea, en color gris, de la ventana de MATLAB, encuentra la opción de *format*. Escribe a continuación el camino que has seguido.

MÓDULO 2: OPERACIONES CON ESCALARES

Alumno: _____ .

1. Haz lo siguiente usando las líneas de comando de Matlab (indicadas por >>)

Operación	*Ejemplo*	*Indica los resultados de las operaciones ejemplo en negrita*
Suma +	>>a=7 >>b=2 **>>suma=a+b** **>>suma**	
Resta -	>>resta=a-b >>otrarresta=7-2 **>>resta** **>>otrarresta**	
Multiplicación *	>>multiplicar=a*b **>>multiplicar**	
División /	>>div1=a/b **>>7/2** **>>div1**	
***División ***	**>>div2=a\b** **>>div3=b\a**	
Potencia ^	**>>a^2**	

2. Ejecuta: >> clear a

(a) Pregunta por el valor de a. ¿Qué sale en pantalla?.

(b) ¿Qué hace el comando CLEAR?.

3. Busca el significado de las siguientes funciones intrínsecas de Matlab, pon un ejemplo y su resultado.

Función	*Significado*	*Ejemplo*	*Resultado ejemplo*
ABS			
SIN			
TAN			
ASIN			
SINH			
EXP			
LOG			
LOG10			
REM			
ROUND			
SQRT			
PI			

MÓDULO 3: GRÁFICAS

Alumno: .

1. **(a)** Busca el significado de las órdenes LINSPACE , PLOT, GRID, DISP, XLABEL y TITLE.
(b) Ejecuta las siguientes instrucciones en la línea de comandos (indicada por >>):

```
>> x=linspace(0,2*pi,30)
>> y=sin(x)
>> disp('Voy a dibujar funciones sinusoidales')
>> plot(x,y)
>> plot(x,y,x,cos(x))
>> plot(x,y,x,y,'y*',x,cos(x),x,cos(x),'r+')
>> grid
>> xlabel('Variable independiente')
>> ylabel('Variables dependientes')
>> title('Primer ejemplo de gráficas')
```

Tras terminar este conjunto de sentencias obtenemos una gráfica, dibújala:

(c) ¿Qué pasaría si añadimos un punto y coma (;) al final de las dos primeras instrucciones indicadas en el apartado (b)?.

2. Dibuja un gráfico de la función tangente con argumento entre los valores de 0 y $\pi/4$. Escribe las sentencias.

3. (a) Busca el significado de MESHGRID y MESH.

(b) Ejecuta el siguiente ejemplo en líneas de comando e indica qué hace:

X3d=-4:0.2:4; Y3d=X3d; [X,Y]=meshgrid(X3d,Y3d); Z=X.*exp(-X.^2-Y.^2); Mesh(X,Y,Z);	

(c) Indica qué hace la primera sentencia del apartado anterior.

(d) Podrías obtener lo mismo con el comando LINSPACE, índica cómo.

(e) Representa la función $z=\exp[-(x^2+y^2)]$ en el dominio [-2,2]x[-2,2]. Indica a continuación las sentencias que has ejecutado.

(f) ¿Por qué sólo es válida la sentencia Z=X.*exp(-X.^2-Y.^2) y no lo es la sentencia Z=X.*exp(-X^2-Y^2)?. Para responder a esta pregunta puedes buscar información, por ejemplo, al final del módulo 4.

MÓDULO 4: VECTORES, MATRICES Y FUNCIONES MATRICIALES

Alumno: .

1. Para asignar valores numéricos a un vector lo podemos hacer como se ha indicado en el módulo anterior o especificando dichos valores en una línea de comandos de Matlab. Indica las salidas que se obtienen tras ejecutar las siguientes órdenes o específica la orden a realizar.

```
>> X=[ 1 2  9 4  5]
```

```
>> X=[1,2,9,4,5]
```

```
>> X=[ 1;2;9;4;5]
```

```
>>X'
```

```
>>X(3)
```

```
>>X([1,3,5])
```

```
>>X(1:4)
```

```
>>X=[1:2:20]
```

Teniendo en cuenta la orden anterior construye un vector X desde el elemento inicial igual a 1 hasta el final igual a 20 equiespaciados en una unidad.

Usando el comando LINSPACE crea un vector X de 20 elementos equiespaciados desde el valor 0 hasta el valor 10.

Genera un vector X cuyo elemento inicial sea el 10 hasta 0 decreciendo en una unidad.

2. Para asignar valores numéricos a cada uno de los elementos de una matriz que denominaremos A se debe ejecutar en la línea de comandos de Matlab, por ejemplo, lo siguiente:

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

¿Cuál es la salida que nos proporciona MATLAB?.

Para saber el valor numérico de un determinado elemento de la matriz A se debe especificar, entre paréntesis y separados por una coma o un blanco, el orden de la fila y columna del elemento deseado. Por ejemplo como en el caso indicado el valor del elemento correspondiente a la primera fila y segunda columna.

```
>>A(1,2)
```

¿Es correcto el resultado?.

Si al valor numérico de este elemento de la matriz A se lo queremos asignar a un dato escalar podemos hacer:

```
>>a12=A(1,2)
```

Ejecuta `>> a(1,2)`

¿Por qué te indica que la variable **"a"** no está definida?.

Para conocer el tamaño, número de filas y de columnas, de la matriz A, hay que ejecutar la sentencia:

```
>> [n,m]=size(A)
```

¿Qué te indica la variable **n**?. ¿Qué te indica la variable **m**?.

Ejecuta, ahora, `>> size(A)`
¿Cuál es la salida y por qué?.

Ejecuta: `>>Z=diag(A)`
Pregunta, tras ejecutar esta última sentencia, por el valor de Z. ¿Z será una matriz, un vector o un escalar?. ¿Qué elementos contiene?

¿Cuál es el resultado de ejecutar `>>A(:,1:2)` ?

¿Cuál es el resultado de ejecutar `>>B=[A A]` ?

¿Cuál es el resultado de ejecutar `>>M=A(:)` ?

¿Cuál es el resultado de ejecutar >>max(A(:)) ?

¿Cuál es el resultado de ejecutar >>**A'** ?

¿Cuál es el resultado de ejecutar >>**A(1,:)** ?

¿Cuál es el resultado de ejecutar >>**A(:,1)** ?

Para conocer el valor del determinante de una matriz cuadrada hay que ejecutar la orden **>>det(A)**

Ejecuta esta orden e indica el resultado

Para determinar la matriz inversa de una matriz cuadrada hay que ejecutar la orden **>>inv(A)**

Ejecuta esta orden e indica el resultado

Para determinar el mayor valor singular de la matriz A se ejecuta >>norm(A)
Ejecútalo para nuestro caso

¿Qué resultados obtenemos y por qué cuando queremos saber el valor del determinante y la inversa de la matriz, siendo B=[1 2 3;4 5 6;7 8 9]?

¿Cuál es el resultado de ejecutar >>**B([1,2],[2,3])** ?

3. Haciendo uso del comando HELP conoce el significado de las siguientes instrucciones de MATLAB y pon un ejemplo con una matriz 3x3.

Función	***Ejemplo en línea de comandos y resultado***
ZEROS	
ONES	
EYE	
TRIU	
TRIL	
COND	
RCOND	

4. Realiza lo que se te indica en los siguientes subapartados haciendo uso de los conocimientos del apartado 2.

(a) Construye una matriz 3x3, pero cuyos elementos sean todos igual a 1. Indica cómo.

(b) Construye una matriz 3x3, pero donde los elementos de la diagonal sean igual a 1 y el resto igual a 0. Indica cómo lo has hecho.

5. Asigna valores numéricos a los elementos de una matriz de tres dimensiones denominada A. Posteriormente, a una matriz B asígnale los siguientes valores numéricos: Primera fila: 1,2,3. Segunda fila: 4,5,6. Tercera fila: 7, 8,9.

Indica los resultados de las siguientes operaciones, explicando el significado de la operación:

>> A*B	**Significado:**	**Resultado:**
>> A .*B		
>> A+B		

>> A-B		
>> A/B		
>> A\B		
>> A.^2		

6. Vamos a encontrar la solución del siguiente sistema de ecuaciones que puede representarse por Ax=b con una función interna de MATLAB:

$$\begin{aligned} x &= 1 \\ 2x+3y &= 2 \\ x+2y+3z &= 4 \end{aligned}$$

(a) Indica cómo asignar en una sola línea de comandos la matriz **A** y el vector **b**

(b) Ejecuta

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

(c) Ejecuta
>>bsol=A*x

(d) Considera ahora el siguiente sistema:

$$\begin{aligned} x+2y+3z &= 1 \\ 4x+5y+6z &= 2 \\ 7x+8y+9z &= 4 \end{aligned}$$

Ejecuta la sentencia dada en el apartado (b), ¿qué ocurre y por qué?.

(e) ¿Qué líneas de comando ejecutarías para poder dibujar las siguientes dos rectas?

5x+7y =3
7x+10y =2

Dibuja la figura que obtienes.

7. Vamos ahora a ver cómo se pueden mostrar vectores en pantalla, para ello vamos a usar el comando **disp**.

(a) Ejecuta e indica el resultado en la columna en blanco:

```
>>a=0
>>x=[1 2 3]
>>y=[4 5 6]
>>disp([a x y])
```

(b) Ejecuta e indica el resultado en la columna en blanco:

```
>>disp([x'  y'])
```

(c) Ejecuta e indica el resultado en la columna en blanco:

```
>>disp([a  x'  y'])
```

MÓDULO 5: CREACIÓN DE UN FICHERO .m

Alumno: .

1. Creación de un fichero .m (o cualquier otro fichero en el sistema MATLAB).
Para crear un fichero .M se debe hacer lo siguiente:

- Pinchar File en el menú de ventanas de MATLAB.
- Seleccionar la opción New y M-file (con ello entraremos en el editor de MATLAB).

(a) Edita el siguiente fichero:

```
function grafica
% Creación de una figura de la función seno entre 0 y 2π.
x=linspace(0,2*pi,30);
y=sin(x);
plot(x,y,x,y,'r+')
grid
xlabel(' x')   %Crear la etiqueta del eje X
ylabel('sen(x)')
title('Gráfica de la función seno')
```

Vete a la ventana Archivo y selecciona Guardar como. Graba el nombre de este fichero en el disco A con el nombre ***grafica.m***

***IMPORTANTE*: El nombre del fichero debe ser el mismo que aparece en la línea *function*. La extensión .m es obligatoria.**

(b) Para saber lo que hace este fichero o programa ejecuta:

```
>> help grafica
```

¿Qué hace el programa ***grafica.m*** que has editado y grabado?.

Por tanto, ¿cuál es el significado del % en el editor de Matlab?.

¿En qué directorio esta guardado?. Recuerda el módulo 1.

(c) Para ejecutar este programa haz,

```
>> grafica
```

Escribe el resultado o resultados de la ejecución.

2. Partiendo del programa ***grafica.m***, realiza las siguientes modificaciones:

- Sustituye la primera sentencia, *function grafica*, por *function grafica2(x)*
- Grábalo con el nuevo nombre de ***grafica2. NOTA IMPORTANTE:*** No lo grabes como grafica2(x).
- Añade al principio el símbolo % a la primera columna de la sentencia número 3 del programa.
- Graba este nuevo programa con el nombre de ***grafica2***

Ejecuta ahora el siguiente conjunto de instrucciones en la línea de comandos de Matlab:

```
>> x=linspace(0,2*pi,30);
>> grafica2(x)
```

¿Obtienes los mismos resultados que con el programa grafica.m?.

Explica el significado de las dos modificaciones que hemos introducido en este nuevo programa grafica2(x) respecto al programa grafica.

3. Crea y graba un programa llamado grafica3 que construya gráficamente la función $\mathbf{3.x^2-e^x}$ en el intervalo [-5,5].

Escribe tu programa

¿Cuántas raíces tiene esta función según puedes observar en la figura que has construido?. ¿En qué intervalos se encuentran?

4. Crea y graba un programa llamado ***medesv.m*** que contenga las siguientes sentencias:

```
function [media,desviacion] = medesv(x)
% Da el valor medio y la desviacion estandar del vector x.
n = length(x);
media = sum(x) / n;
desviacion = sqrt(sum((x - media).^2)/n);
```

¿Cuál es el significado de las órdenes SUM y LENGTH?

Ejecuta las sentencias:

```
>>x=[1 2 3 4 3 2 1]
>>[MEDIA,DESVIACION]=medesv(x)
```

Escribe los resultados.

Ejecuta:

```
>>MEDIA
>>DESVIACION
```

Escribe los resultados.

Ejecuta:

```
>> MEDIA=1000.0; DESVIACIÓN=500.0;
>> medesv(x)
```

Pregunta ahora por el valor de MEDIA Y DESVIACIÓN. ¿Qué ha ocurrido?.

Ejecuta:

```
>> MEDIA=1000.0; DESVIACIÓN=500.0;
>> [MEDIA, DESVIACION]=medesv(x)
```

Pregunta ahora por el valor de MEDIA Y DESVIACIÓN. ¿Qué ha ocurrido?.

Añade como variable dentro del programa y como variable de salida, además de media y desviación, una variable llamada SUMA. Esta variable debe almacenar el valor de la suma de todos los elementos del vector x, también llamada variable de entrada. Graba y llama a este nuevo programa ***medevsum.m.***

Escribe cómo lo has hecho y ejecútalo.

5. Crea, y escribe a continuación, un programa llamado ***sistema.m*** que resuelva el sistema de ecuaciones

$$\begin{aligned} x &= 1 \\ 2x+3y &= 2 \\ x+2y+3z &= 4 \end{aligned}$$

considerando a la matriz A y el vector b variables de entrada y al vector solución x variable de salida.

6. Edita y ejecuta un programa, llámalo ***norma.m,*** que determine la norma euclídea de un vector.

MÓDULO 6: OPERACIONES E INSTRUCCIONES RESERVADAS EN MATLAB

Alumno: .

A continuación tienes un resumen de operaciones e instrucciones del lenguaje de programación de Matlab que te pueden ser útiles para entender los programas que vas a editar y crear.

BUCLES (Permiten que un conjunto de órdenes se repitan un número de veces predeterminado)

	Estructura
FOR	FOR var = vector Conjunto de sentencias END
WHILE	WHILE condición Conjunto de sentencias que se ejecutan mientras se cumpla la condición END

1. Ejecuta el siguiente ejemplo de FOR que permite estimar $\sum_{i=1}^{n} i^2$

```
>> serie=0; n=10; FOR i=1:1:n; serie=serie+i*i; end
```

¿Cuál es el resultado que obtienes?

Construye un programa llamado ***sumaserie*** que tenga como variable de entrada la variable n y como variable de salida la variable serie. Escribe el programa y cuáles serian las órdenes en Matlab que darías para ejecutarlo correctamente.

Programa

Órdenes en Matlab

```
>>
>>
>>
>>
```

2. Edita y ejecuta el siguiente programa, grabándolo como *seno*, y di lo que hace:

```
function seno
FOR n=1:10
  x(n)=sin(n*pi/10)
END
```

3. Ejecuta el siguiente programa, grábalo como *matriz.m*, y di lo que hace:

```
Function [A]=matriz
FOR n=1:3
  disp(n)
 FOR m=3:-1:1
    disp(m)
    A(n,m)=n^2+m^2
   pause
 END
END
```

4. Construye y ejecuta un programa llamado *factorial.m* que me permita determinar el factorial de un número N, donde N será una variable de entrada y FACTOR la variable de salida. Escribe el programa y las órdenes en Matlab que debes dar para que el programa se ejecute correctamente. *NOTA: puedes usar el comando PROD.*

5. Ejecuta el siguiente ejemplo de WHILE en líneas de comando de Matlab:

```
>>delta=1
>>WHILE (1+delta)>1;
      delta=delta/2
   END
>>delta=2*delta
```

Comprueba si el resultado que obtienes coincide con la función interna EPS de Matlab. *Nota: Ten en cuenta la lección de aritmética del computador.*

Edita y ejecuta el programa que contenga las sentencias del ejemplo anterior. Grábalo como *epsilon.m*, considerando que la variable delta sea un variable de salida. ¿Cómo lo ejecutarías con líneas de comando en Matlab?.

Ejecuta las siguientes sentencias

>>delta=1; num=0; >>WHILE (1+delta)>1; num=num+1 delta=delta/2 END >>delta=2*delta	¿Qué nos proporciona la variable **num**?

OPERACIONES RELACIONALES Y LÓGICAS

Operaciones RELACIONALES		***Operaciones LÓGICAS***	
<	Menor que	&	AND
<=	Menor o igual que	|	OR
>	Mayor que		
>=	Mayor o igual que		
= =	Igual que		
~ =	No igual		

ESTRUCTURAS IF-ELSE-END

Estructura
IF condición Sentencias END
IF condición Ordenes a ejecutar si la condición es verdadera ELSE Ordenes a ejecutar si la condición es falsa END

6. Edita el siguiente programa, llámale ***triang.m***, y ejecútalo

```
function result=triang(a,b,c)
% Nos dice si con las variables de entrada a,b,c
% pueden formar un triangulo equilatero
% Ordenamos los lados de menor a mayor
x=sort([a,b,c]);
if x(3)==x(2) & x(3)==x(1)
   result='Equilatero';
else
   result='No es equilatero';
end
```

La variable *result* es lo que se conoce como variable alfanumérica o variable de tipo carácter. Con este tipo de variables no es posible realizar operaciones aritméticas ya que no es una variable numérica. La asignación de los caracteres se ha de realizar entre apóstrofes.

¿Cuál es la diferencia que encuentras cuando eliminamos todos los puntos y comas de las sentencias?

¿Qué obtienes si haces `>>help triang`?

7. Considerando el programa que hiciste en el apartado 5 del módulo 5, incorpórale que cuando el determinante de la matriz A sea cero no se ejecute el programa y nos de una indicación en pantalla que nos diga que el sistema está mal condicionado. Índica a continuación cómo lo has hecho.

8. Edita y ejecuta un programa llamado ***ejemif.m,*** que tenga como variable de entrada, x, que sea numérica y escalar y como variable de salida, xsal, que contenga la operación de dividir x entre 10 si x es mayor o igual que 5 y en caso contrario que nos informe a través de una variable alfanumérica que x es menor que 5 y que xsal sea el cuadrado de x.

9. Ejecuta y graba el siguiente programa. Este programa tiene errores, tras tu análisis escribe en el recuadro en blanco un programa ***ejemif2.m*** que sea correcto.

```
Function [C0,C1]=ejemif3(M)
IF M < 0
   Disp('No puede ser M negativo')
  C0=-999.9;
  C1=-999.9;
   Break
ELSE
  IF (M >= 0) & (M <= 5)
    Disp('M positivo y menor o igual que 5')
    C0=(1-10/100)*C;
    C1=M
  ELSE
    Disp('M mayor que 5')
    C0=(1-20/100)*M;
    C1=M;
END
```

(b) Indica cómo ejecutarlo en líneas de comando.

© ¿Qué hace el comando BREAK?

MÓDULO 7: ARITMÉTICA DEL COMPUTADOR

Alumno: .

Mira tus apuntes sobre este tema

1. (a) Comprueba usando Matlab que el computador que utilizamos para prácticas sigue la representación IEEE754 para representar números reales, para ello utiliza sólo la base y el exponente.
NOTA: Matlab trabaja con números en doble precisión en representación IEEE754

(b) Comprueba que la función interna *eps* indica la precisión de los cálculos y que debe ser igual a 2^(-52).

(c) Ejecuta 2^(-53)+1. Por qué obtenemos el resultado de 1

(d) ¿Cuál es el número más pequeño que podemos representar que sea distinto de cero?.

(e) Ejecuta las dos sentencias, indicando por qué esos resultados, 10^308 y 10^309

2. El siguiente programa también permite saber la precisión de un computador :

```
function precis
 % Cálculo de la precision o epsilon del computador
 k=1;
 while ((1.0+2^(-k))>1.0)
        k=k+1;
 end
 %Ver los resultados
 disp ('Numero de bits de la mantisa : '); disp (k-1)
 disp('Precision del computador : ');  disp (2^(-(k-1)))
```

Edita y ejecuta este programa. Escribe los resultados que obtienes he indica si son o no correctos.

3. El siguiente programa nos permite redondear al número de cifras que se desee, n, un determinado número x; teniendo en cuenta que el redondeo del número de cifras incluye la parte entera y la parte decimal del número.

```
function y=redondeo(x,n)
if nargin~=2; error('Numero incorrecto de argumentos');end
m=floor(log10(abs(x))); %Notación científica
%Escalado para la parte entera
escala=10^(n-m-1);
%Obtencion del valor
y=round(escala*x)/escala;
```

Con el comando help puedes conocer las funciones intrínsecas que desconozcas, como por ejemplo *nargin*.

(a) Ejecuta este programa para x=pi y para x=pi/6 para confirmar lo que se te indica en el enunciado, anotando los resultados para ambos casos.

(b) Veamos ahora como se redondean y propagan los errores en operaciones elementales como +, -, * y /. Para obtener el error cometido en la suma ejecuta lo siguiente:

>>format long >>x=pi; >>xa=redondeo(x,4) Indica los resultados	>>format long >>x=pi*10 >>xa=redondeo(x,4) Indica los resultados	>>format long >>x=pi*100 >>xa=redondeo(x,4) Indica los resultados	>> y=(22/7)*10^(-6) >> ya=redondeo(y,4) Indica los resultados

¿Qué hace la función redondeo?

(c) Ejecuta el siguiente programa *redond.m*:

Function redond
 	
Format long
x=pi;
y=22/7;
n=4;
 	
%Redondeo siempre los números considerando n dígitos
xa=redondeo(x,n); ya=redondeo(y,n);
suma_a=(xa+ya);
suma_redondeo_a=redondeo(xa+ya,n);
resta_a=(xa-ya);
resta_redondeo_a=redondeo(xa-ya,n);
multip_a=(xa*ya);
multip_redondeo_a=redondeo(xa*ya,n);
div_a=(xa/ya);
div_redondeo_a=redondeo(xa/ya,n);
 	
r=suma_a-suma_redondeo_a;
re=resta_a-resta_redondeo_a;
rm=multip_a-multip_redondeo_a;
rd=div_a-div_redondeo_a;
	Indica en este espacio los errores relativos que obtienes de las operaciones suma, resta, multiplicación y división:
	
	¿Con que dos operaciones anteriores se comete mayor error?.

```
% Sin redondear
suma=x+y;
resta=x-y;
multip=x*y;
div=x/y;

% Errores
error_abs_suma=suma-suma_redondeo_a;
error_rel_suma=abs(error_abs_suma/suma);
error_abs_resta=resta-resta_redondeo_a;
error_rel_resta=abs(error_abs_resta/resta);
error_abs_multip=multip-multip_redondeo_a;
error_rel_multip=abs(error_abs_multip/multip);
error_abs_div=div-div_redondeo_a;
error_rel_div=abs(error_abs_div/div);
disp('  x  y  SUMA  xa  ya  redondeo_a error_abs error_rel')
disp([x  y  suma])
disp([xa  ya  suma_redondeo_a])
disp([error_abs_suma error_rel_suma])
disp('  x  y  RESTA  xa  ya  redondeo_a error_abs error_rel')
disp([x  y  resta])
disp([xa  ya  resta_redondeo_a])
disp([error_abs_resta error_rel_resta])
disp('  x  y  MULTIPLICAR  xa  ya  redondeo_a error_abs error_rel')
disp([x  y  multip])
disp([xa  ya  multip_redondeo_a])
disp([error_abs_multip error_rel_multip])
disp('  x  y  DIVISION  xa  ya  redondeo_a error_abs error_rel')
disp([x  y  div])
disp([xa  ya  div_redondeo_a])
disp([error_abs_div error_rel_div])
```

Indica en este espacio los errores relativos que obtienes de las operaciones suma, resta, multiplicación y división considerando 5 dígitos de redondeo.

Indica en este espacio los errores relativos que obtienes de las operaciones suma, resta, multiplicación y división considerando 3 dígitos de redondeo.

4. Ejecuta el siguiente programa que nos va a permitir ver el condicionamiento de una función, en este caso de un polinomio de grado 10. Es importante conocer si una función está mal o bien condicionada, ya que en caso de que esté mal condicionada, como en este ejemplo, los resultados que obtendríamos serían poco fiables. Un método para saber si un polinomio está mal condicionado es alterar levemente alguno de sus coeficientes.

```
Function ejpoly
X=[1:10];  %X son las raices de un polinomio de grado 10
P=poly(X); % Calculamos el polinomio
P(2)=P(2)+0.001; %Modificamos el polinomio
%Obtenemos las nuevas raices, distintas de X reales.
Roots(P)
```

Indica cuáles eran las raíces X y las nuevas raíces tras la modificación del polinomio

5. Un algoritmo es inestable cuando los errores de redondeo se acumulan degradando la exactitud del resultado final, a pesar de que en aritmética exacta el algoritmo sea correcto. La sucesión siguiente es un ejemplo de algoritmo numérico inestable,

$$x_{n+2} = \frac{13}{3} x_{n+1} - \frac{4}{3} x_n, \qquad \textit{tomando como valores iniciales } x_0 = 1 \textit{ y } x_1 = \frac{1}{3}$$

Se puede comprobar que el término general de esta sucesión es igual a:

$$x'_n = \left(\frac{1}{3}\right)^n = \frac{1}{3} x'_{n-1}$$

Si suponemos que x'_n son los valores exactos de la sucesión se puede estimar el error de redondeo de x_n. Escribir un programa en MATLAB ***(.m)*** que resuelva esta sucesión y visualice en pantalla, desde n igual a 1 hasta 20 iteraciones, x_n, x'_n, el error absoluto $\left|x_n - x'_n\right|$ y el error relativo $\left|\frac{x_n - x'_n}{x'_n}\right|$. Escribe el programa y los resultados que obtienes.

6. Ejecuta el siguiente programa, considerando como variable de entrada el vector de 1 elemento x=[2^(10/2)]

```
function S=norma2(x)
S=0;
S=S+(x^2);
S=sqrt(S);
```

(a) ¿Qué resultado obtienes para la variable de salida S?.

(b) Ejecuta, ahora, la función interna de Matlab *norm(x,2)*?. ¿Cuál es el resultado?. ¿Qué hace esta función interna para vectores?

(c) Ejecuta la función interna norm(x,2) para el vector x=[2^(1024/2)]. Indica el resultado.

(d) Ejecuta la función norm2(x) para el vector x=[2^(1024/2)]. Indica el resultado. ¿Por qué obtienes este resultado?.

MÓDULO 8: SISTEMA DE ECUACIONES

Alumno: .

1. Analicemos el condicionamiento de una matriz con MATLAB. Matlab incorpora funciones para conocer el condicionamiento de un sistema de ecuaciones a través de las funciones intrínsecas ***cond***, ***rcond*** o ***condest.***
Haciendo uso de las tres funciones anteriores y de la función ***flops***, escribe un programa llamado ***condicion.m*** que nos diga los valores de los tres números de condición, del determinante de la matriz y el número de operaciones consumidas en el cálculo de cada uno de ellos para el sistema:

$$0.832\ x_1 - 0.448\ x_2 = 1$$
$$0.784\ x_1 + 0.421\ x_2 = 0$$

Escribe, a continuación, la función condicion.m, y los resultados de su ejecución. Considera la matriz A del sistema y el vector independiente b como variables de entrada.

2. Realiza un programa en Matlab que resuelva un sistema de ecuaciones triangular inferior. Llama a este programa ***triainf.m***. Da la solución del sistema:

$$x = 1$$
$$2x+3y = 2$$
$$x+2y+3z = 4$$

3. El siguiente programa permite obtener la matriz triangular inferior según la factorización de Cholesky.

```
function L= factchol(A)
% Factorización de Cholesky
% Valido para matrices simétricas cuadradas y definidas positivas.
% A = matriz simetrica y positiva
% L = es una matriz triangular inferior en L, tal que A = L*L'.

if any(any(A-A'))~=0, error('la matriz no es simetrica');end

[n,m]=size(A);if n~=m; error('Matriz no cuadrada');end

for i=1:n
  if A(i,i)<=0
    error('La matriz no es positiva');
  end
end

H(1,1)=sqrt(A(1,1));
A(1,1)=H(1,1);

for k=2:n
  for i=1:k-1
    suma=0;
    for j=1:i-1
      suma=suma+H(i,j)*H(k,j);
    end
  H(k,i)=(A(k,i)-suma)/H(i,i);
  A(k,i)=H(k,i);
  end
  suma2=0;
  for jl=1:k-1
    suma2=suma2+H(k,jl)*H(k,jl);
  end
  H(k,k)=sqrt(A(k,k)-suma2);
  A(k,k)=H(k,k);
end

L=tril(A);
```

(a) Busca el significado de ***any*** y de ***error*** y escríbelo a continuación.

(b) Ejecuta este programa para la matriz A=[4 2 2 4;2 5 7 0;2 7 19 11;4 0 11;25]. Indica el valor de L y confirma que es correcto indicando cómo lo harías en una sentencia de comandos

(c) Confirma con la función intrínseca ***chol*** que su matriz salida es igual que la matriz L de nuestro programa. Indica cómo lo has hecho.

(d) Ejecuta las siguientes líneas de comando para obtener una nueva matriz A, buscando el significado de las funciones internas de Matlab que desconozcas.

```
>> rand('seed',sum(100*clock));
>> n=3;
>>A=rand(n);
>>A=(A+A')/2+eye(n);
```

Aplica de nuevo el programa *factchol.m* y la función interna *chol* y confirma que los resultados obtenidos son correctos.

(e) Si A es una matriz simétrica y positiva sabemos que **A=L*L'**, por tanto un sistema de ecuaciones en donde intervenga la matriz A lo podemos descomponer en **(L*L').x=b,** siendo x el vector solución y b el vector independiente.

Para obtener el vector solución x podemos descomponer este sistema en dos:

L.u=b

L'.x=u

Considerando lo anterior, la matriz A generada en el apartado (d) y como vector independiente b=[1;2;3]
Escribe a continuación las sentencias y los resultados que obtienes utilizando el programa ***factchol.m***

Sentencias en líneas de comando de Matlab	>> >> >> >> >> >> >> >>
Vector u	
Vector x	
Confirma en una línea de comando que x está bien	>>

4. Edita el programa ***jacobi1.m*** con el siguiente conjunto de sentencias:

```
function [x]=jacobi1(A,b,x0)
%
% x=jacobi1(A,b,x0). Aplica UNA iteración del método iterativo de Jacobi
%                    al sistema de ecuaciones A . x = b, usando x0 como valor
%                    de la aproximación a la solución en la iteración anterior.
% Valores de entrada: A (matriz con los coeficientes del sistema)
%       b  (vector COLUMNA con los términos independientes del sistema)
%       x0 (vector COLUMNA con la aproximación a la solución en la iteración anterior)
% Valores de salida: x (nueva aproximación a la solución)
%
L=A-triu(A)
U=A-tril(A)
D=diag(diag(A))
x=inv(D)*(b-(L+U)*x0)
```

(a) Utilizando ***jacobi1.m***, determina la primera solución del sistema de ecuaciones:

3x+y+z=4
2x+5y+1x=-1
-x+y+3z=4 partiendo de (0,0,0) como aproximaciones iniciales de la solución.

Tras la asignación de los valores numéricos a las variables, escribe la sintaxis de la función:

```
>> [h]=jacobi1(A,b,x0)
```

¿Qué resultado obtienes para h?. ¿Cuál es el significado de h?.

5. Escribe una función ***x=jacobi(A,b,x0,tol)*** que resuelva el sistema de ecuaciones **A.x=b** por el método **iterativo de Jacobi** hasta conseguir un error de convergencia inferior a la tolerancia *tol*, partiendo de un valor ***x0*** como aproximación inicial a la solución.

La ejecución de la función debe incluir la presentación en pantalla de los pasos o iteraciones sucesivas, mostrando los valores de número de iteración s, los valores de las incógnitas $x_i^{(s+1)}$ (i=1,...n) y el error entre $\mathbf{x}^{(s+1)}$ y $\mathbf{x}^{(s)}$.

Aplicar la función *jacobi.m* al sistema del ejercicio anterior, considerando una tolerancia de 0.001. Comprobar el resultado con tu programa ***sistema.m.***

X =	**Nº de iteraciones :**

Escribe el programa ***jacobi.m***

7. Utilizando el programa ***jacobi.m***, para resolver el sistema de ecuaciones:

x+2y+3z=4
4x+5y+6z=-1
7x+8y+9z=4 partiendo de (0,0,0) como aproximaciones iniciales de la solución.

¿Qué soluciones obtienes?. Si obtienes soluciones incorrectas o no te permite calcular la solución modifica *jacobi.m* para que el programa funcione correctamente.

x =	**Nº de iteraciones :**

Escribe el conjunto de sentencias que has incorporado al programa ***jacobi.m***, especificando sólo entre qué sentencias se encuentra, es decir escribe la línea anterior, conjunto de sentencias nuevas y líneas posterior que tenías de tu programa de jacobi.m.

8.Teniendo en cuenta tu programa de jacobi.m implementa las sentencias necesarias para analizar la convergencia del método de Jacobi para el siguiente sistema de ecuaciones, teniendo en cuenta diferentes valores de β.

2x-2y = 4
2x+3y-z=-1
βx+2z=4

β	***Radio espectral***	***Solución si la hubiera***
-1		
-3		
-5		

Nota*: Utiliza las funciones internas de Matlab* ***eig*** y ***max.***

Escribe el conjunto de sentencias nuevas que has incorporado al programa, especificando también las dos sentencias entre las que se encuentra este conjunto de sentencias.

9. Escribir una función **x=jacobi_amor(A,b,x0,w,tol)** que resuelva el sistema de ecuaciones **A**·**x** = **b** por el método iterativo Jacobi amortiguado, hasta conseguir un error de convergencia inferior a la tolerancia *tol*, partiendo de un valor **x0** como aproximación inicial a la solución y *w* como peso.
Nota: Con añadir en tu programa jacobi.m la sentencia de amortiguamiento tras la sentencia que define el método de jacobi es suficiente.
¿El Análisis de Convergencia que hiciste en el problema 8 es correcto para este método amortiguado?
Aplica tu programa al sistema de ecuaciones del ejercicio 8 para un valor β que dé la solución con una tolerancia de 0.001, un peso de 0.9 y vector solución inicial [0;0;0].

W=0.9	**Nº iteraciones =**	**Vector solución x =**

Escribe, a continuación, la función.

10. Supongamos el siguiente sistema de ecuaciones:

$$\begin{bmatrix} -4 & 1 & 1 & 1 \\ 1 & -4 & 1 & 1 \\ 1 & 1 & -4 & 1 \\ 1 & 1 & 1 & -4 \end{bmatrix} \vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Construye un solo programa, llámale ***elegir.m***, que permita, según el valor numérico que le asignemos a una variable de entrada que denominaremos m, se ejecute el método de Jacobi cuando m=1 o el método de Jacobi amortiguado cuando m=2 una tolerancia igual a ±0.00001 y con una aproximación inicial a la solución (0 0 0 0).
Como variables de entrada: la matriz A, el vector b, la tolerancia, el vector solución inicial, el peso w y m.
Como variables de salida tendrás el número de iteraciones y el vector solución.
NOTA: Si m, variable de entrada al programa, es distinta a los dos valores indicados que opte por m=1 y que te indique en pantalla lo que has hecho. Usa el comando pause antes de que se realice la ejecución para poder verlo.
Escribe el programa ***elegir.m.***

MÓDULO 9: RAICES DE UNA FUNCIÓN DE UNA VARIABLE

A partir de aquí y siguiendo una estructura similar dada en los módulos anteriores y en este módulo construye tus propios informes de prácticas.

¿Cómo calcula MATLAB las raíces?

MATLAB proporciona la función x=***fzero***(nombre_funcion,x0,tol,it) para obtener la raíz de una función. La función que ha de ser de la forma y=nombre_funcion(x) se introduce como primer argumento en fzero y entre apóstrofes, x0 es la aproximación inicial, el número de iteraciones del proceso iterativo para alcanzar la solución se introduce por medio del argumento it. Si it es igual a 1, el proceso se repite hasta que la solución esté dentro de una tolerancia tol. Los dos últimos argumentos se pueden omitir. Esta función ***fzero*** emplea el método de Brent que combina la interpolación cuadrática inversa con la bisección.

¿Cómo calcula MATLAB las raíces de un polinomio?

La función MATLAB r=***roots***(c) calcula las raíces r de un polinomio cuyos coeficientes se encuentran almacenados en el vector c de la forma: $c(1)x^n+...+c(n)x+c(n+1)$.
Otra función relacionada con los polinomios es c=***poly***(r) que genera un polinomio con las raíces introducidas como argumento en el vector r.

1. Sea la siguiente función externa ***bisec.m*** que permite calcular raíces de una función por el método de la bisección.

```
function [n,x,A]=bisec(nombre,a,b,ex,ey)
if nargin<3;error('numero insuficiente de argumentos'); end
if nargin<4;ex=eps;end
if nargin<5;ey=eps;end
ya=feval(nombre,a);
yb=feval(nombre,b);
if ya*yb>0; error('No se cumple Bolzano'); end
% n - Nº de iteracciones que debemos hacer considerando |f(raiz)|<tolerancia
n1=ceil((log(abs(b-a))-log(ex))/log(2));
n=n1+1
x=zeros(n,2);A=zeros(n,2);

for k=1:n
m=(a+b)/2;
ym=feval(nombre,m);
x(k,1)=m;
x(k,2)=ym;
A(k,:)=[a,b];
if abs(ym)<ey | abs(b-a)<ex;break;end
if ya*ym>0
   a=m;
   ya=ym;
else
   b=m;
   yb=ym;
end
end
if k<n
  x=x(1:k,:);
  A=A(1:k,:);
end
```

(a) Busca el significado de feval, nargin y ceil.
(b) Edita y graba la función y=enlfx(x) que genere la función **$f(x)=2.\cos(2x)-(x+1)^2$**
Ejecuta las siguientes sentencias para ver donde están las raices de f(x) en el intervalo [-6,6]

```
>> z=linspace(-6,6,50);fz=enlfx(z);
>> plot(z,fz);grid on;
```

Pinta la figura en papel.
(c) Ejecuta el programa ***bisec.m*** de la forma

```
>>[n,x,A]=bisec('enlfx',-4,-2,0.01,0.01)
```

NOTA: Observa que la función enlfx está encerrada entre apóstrofes.
¿Cuál es el significado de a, b, ex y ey y sus valores numéricos?.
Indica los valores numéricos de n, x y A y que contienen.
¿Cuál es la raíz de esta función en el intervalo [-4,-2]?

2. Sea la siguiente función ***bisect.m***

```
function bisect(f,a,b,epsilon)
%BISECT Metodo de la Biseccion para f(x)=0.

format short;
disp([' iter a b fa fb c fc']);
it=0;
fa=feval(f,a);
fb=feval(f,b);

if fa*fb >= 0
  error('fa*fb no es negativo!')
  break
end
  c = (a+b)*0.5;
  fc = feval(f,c);

while abs(fc) > epsilon
  c = (a+b)*0.5;
  fc = feval(f,c);
  it=it+1;
  disp([it,a,b,fa,fb,c,fc])
  if fc==0
    return
    end
  if fa*fc<0   %raiz a la izquierda del punto medio
    b=c;
    fb=fc;
  else         %raiz a la derecha del punto medio
    a=c;
    fa=fc;
  end
end
c=(a+b)*0.5;
error=feval(f,c);
```

(a) Ejecuta este programa para f(x) del apartado anterior dentro del intervalo [-4,-2].
(b) Indica los resultados que obtienes
(c) Modifica la función anterior para que te dé como variables de salida el número de iteraciones realizadas, la raíz y el valor de la función en la raíz.

(d) Aplicar la función *bisect.m* a **$f(x)= (x-1).e^{-x}$,** considerando una tolerancia de 0.001 en el intervalo [0,3]. Ayúdate del comando PLOT para confirma que la raíz se encuentra dentro de este intervalo.
Comprobar el resultado con el obtenido mediante la función intrínseca de MATLAB llamada FZERO.
Indica cómo has ejecutado este programa, la raíz y el número de iteraciones realizadas.

3. Escribir una función [x,it] = **interpol**(fun,tol,x0,x1) que obtenga numéricamente una raiz *x* de la función *fun(x)* usando el método iterativo de interpolación lineal con una tolerancia *tol*, siendo *x0* y x1 los valores iniciales (próximos a la raíz exacta x) con los que comienza el método. La ejecución de la función debe incluir la aparición en pantalla de los pasos o iteraciones sucesivas, mostrando los valores de : nº de iteración, la solución aproximada x y el valor de fun(x)
Aplica tu programa a la función f(x) del ejercicio anterior. No te olvides escribir tu programa.

4. Ejecuta **interpol.m** para $f(x)= x^2-3x+1$ con una tolerancia de 10^{-4} en el intervalo [2,3]. Confirma primero que en ese intervalo existe una raíz. Las soluciones sucesivas que obtienes coinciden con el ejemplo dado en los apuntes.

5. Escribe **newton.m** para encontrar la raíz de la función anterior en el intervalo dado.

MÓDULO 10: APROXIMACIÓN DE UNA FUNCIÓN POR UN POLINOMIO

1. Escribir una función **f1** = difdiv(**x**, **f**, **x1**) que, a partir de *n+1* datos conocidos (x_i,f_i), interpole el valor de una función desconocida f(x) en puntos cuyas abscisas se indican en el vector **x1**, haciendo uso del método de las diferencias divididas. La ejecución de difdiv.m debe incluir la aparición en pantalla de los coeficientes del polinomio de interpolación $P_n(x)$. Los vectores **x** y **f** contienen las abscisas x_i y las ordenadas $f_i = f(x_i)$ respectivamente.

2. La temperatura *t* medida en ciertos puntos *x* próximos a un foco de calor han sido :

Temperatura *t* (ºC)	**8.8**	**7.2**	**6.0**	**4.2**	**2.8**
Punto *x* (m)	**0.5**	**1.0**	**2.0**	**3.0**	**4.0**

(a) Representar gráficamente los cinco datos (x,t) mediante el comando PLOT.
(b) Calcular el polinomio de interpolación $P_4(x)$ por diferencias divididas.
(c) Estimar la temperatura t en posiciones distantes 1.5 y 2.5 m de acuerdo con el polinomio $P_4(x)$.
(d) Comprobar el resultado anterior del apartado (c) con el obtenido con la función intrínseca de MATLAB llamada INTERP1.
La función y1=***interp1***[x,y,x1,'tipo') que produce una interpolación lineal (tipo=linear), cúbica (tipo=cubic) o por splines cúbicos (tipo=splin). Esta función tiene una serie de restricciones (véase help interp1). Hay una función llamada spline que además de interpolar valores por medio del método de los splines cúbicos puede devolver, como parámetro de salida, la función splin.
(e) Superponer en la gráfica del apartado a) el trazo de la curva $P_4(x)$ y los puntos interpolados. Usa el comando HOLD ON para superponer los gráficos.

$P_4(x)$ =		
Punto x (m)	**1.5**	**2.5**
Temperatura t(ºC) según $P_4(x)$		
Temperatura t(ºC) según INTERP1		

3. Busca una función intrínseca de Matlab que nos permita ajustar un conjunto de puntos por mínimos cuadrados.
(a) Aplica esta función a los siguientes datos, para ello sería conveniente que primero visualizaras el conjunto de datos para establecer el mejor ajuste.

X	**4.0**	**4.5**	**5.0**	**5.5**	**6.0**	**6.5**	**7.0**	**7.5**

f(x)	**2.2**	**3.5**	**4.0**	**6.0**	**6.5**	**7.3**	**8.2**	**8.7**

MÓDULO 11: DIFERENCIACIÓN NUMÉRICA DE UNA FUNCIÓN

1. Escribir una función **dn**=**difinc3p**(fun,x0,**h**) que calcule numéricamente la derivada primera de la función fun(x) en el punto x0 utilizando el método de diferenciación directa por diferencias finitas centradas de 3 puntos considerando diferentes valores del intervalo h (reunidos en el vector **h**).

2. Utilizar la función del ejercicio anterior para estimar numéricamente la derivada primera de f(x)=arctg(x) en $x0=(2)^{1/2}$, tomando como h los valores de 1, 0.1, 0.01 y 0.001.

h	**1**	**0.1**	**0.01**	**0.001**
dn				

MÓDULO 12: INTEGRACIÓN NUMÉRICA DE UNA FUNCIÓN

1. Escribir una función [I,h] = **simpson**(fun,a,b,n) que calcule numéricamente la integral de la función fun(x) en el intervalo [a,b] dividido en n subintervalos de longitud h utilizando la fórmula de Simpson cada 2 subintervalos. Considerar el supuesto de que el número n de subintervalos sea impar, aplicando la fórmula del trapecio sobre el último subintervalo.

2. Sea la función $f(x) = \exp(-x^2)$:

(a) Representar la función f(x) en [0,1].

(b) Utilizar la función *simpson.m* para estimar numéricamente la integral I de f(x) en [0,1] con una cota de error inferior a 0.001, indicando el número de subintervalos n considerados.

Solución :	**I =**	**n =**	**h =**	**Cota de error :**

(c) Comparar el resultado anterior de I con el obtenido con la función TRAPZ de MATLAB.

I (obtenido con TRAPZ):