INTRODUCCIÓN AL CÁLCULO NUMÉRICO Y PROGRAMACIÓN

PRÁCTICAS DE PROGRAMACIÓN CON MATLAB

CURSO 2005-2006

GRUPO D

Las prácticas de Introducción de Cálculo Numérico y Programación con MATLAB están divididas en los siguientes módulos:

Módulo	
1	Nociones básicas
2	Representación gráfica, creación de programas.
3	Operaciones e instrucciones reservadas en MATLAB.
4	Repaso
5	Arquitectura del computador. Sistemas de ecuaciones lineales.
6	Resolución de sistemas de ecuaciones lineales. Métodos iterativos.
7	Raíces de una función de una variable.
8	Sistema de ecuaciones lineales.
9	Interpolación y aproximación a funciones.
10	Diferenciación e integración.

Bibliografía utilizada

Cálculo numérico para computación en Ciencia e Ingeniería. Desarrollo práctico con MATLAB. Editorial Síntesis. Ignacio Martín Llorente y Víctor M. Pérez García.

Métodos Numéricos con MATLAB. Editorial Pearson Educación. John H. Mathews, Kurtis D. Fink.

Análisis numérico y visualización gráfica con Matlab . Editorial Pearson Educación. Shoichiro Nakamura.

Introducción

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

- MATLAB se puede arrancar como cualquier otra aplicación de Windows 95/98/NT, presionando dos veces en el icono correspondiente o por medio del menú de inicio. Una vez abierto aparece una consola similar a esta :

```
        This is a Classroom License for instructional use only.
      Research and commercial use is prohibited.
      To get started, select "MATLAB Help" from the Help menu.
>>
```

- Desde la consola de Matlab, colócate en tu directorio de trabajo que en nuestro caso será la unidad A: (tu disquete)

  >> cd A:

- Para grabar el trabajo del día , emplea el comando diary.
- Diary guarda todo el que aparece en la consola durante la sesión MATLAB. Por tanto, nada mas comenzar la clase, teclea:

  >> diary('practicas.txt')

  Al acabar la clase cierra el diario con

  >>diary off

  Si quieres continuar al día siguiente tu diario, teclea

  >> diary('practicas.txt')

  De esta forma continuaras por donde acabaste el día anterior y todo lo que has hecho quedará grabado en un fichero llamado *practicas*
- Al terminar cada sesión se debe comprobar que se graba el trabajo realizado en vuestro disquete personal, disquete A. Desde el explorador de Windows comprueba que está el fichero practicas.txt.
- Para garantizar que se va a trabajar en la unidad A escribir en la línea de comandos:

  >> cd a: % de este modo te sitúas en el interior del disquete

  >> dir % da un listado de los ficheros de tu disquete

  Con ello garantizamos que desde el comienzo de la sesión de prácticas trabajamos con nuestro disco de trabajo.
- Una vez que se ha realizado un módulo, las prácticas obligatorias de dicho módulo se deben enseñar al profesor de prácticas para que sean evaluadas

MÓDULO 1: NOCIONES BÁSICAS

¿CÓMO NAVEGAR POR EL INTERIOR DEL MATLAB?

En general,

1. MATLAB distingue entre mayúsculas y minúsculas, aunque muchos comandos estén definidos tanto en mayúsculas como en minúsculas.

POR LO GENERAL ESCRIBE LOS COMANDOS EN MINÚSCULAS

HELP Y HELPWIN

1. Si quieres conocer el funcionamiento de una determinada herramienta teclea

   >>help "nombre del comando"

   Ejemplo:
   *>> help mean %Te indica como funciona **mean**, que calcula la media de una matriz*

   Un resultado análogo encontrarás al teclear **helpwin**, en el que se te mostrará la ayuda en una ventana separada y en un formato más estético.

   >> helpwin mean

2. Si deseas conocer **toda** la ayuda del matlab
   >> help
   help (o helpwin) te guiará por el interior del matlab para que localices las funciones que quieres emplear.
   Ejemplo:
   si quieres utilizar algún comando para trabajar con matrices
   >> help
   Ahí te enumeran una serie de directorios, en uno de los cuales pone
   matlab\elmat - Elementary matrices and matrix manipulation
   Si tecleas ahora
   >> help matlab\elmat
   te aparecera una lista de funciones de matlab . Si quieres saber mas acerca de cada una de ellas, averigualo con el help `. Así por ejemplo
   >> help zeros

3. Puedes saber más sobre la Ayuda haciendo uso del interfaz gráfico del que dispone **MATLAB**. En la parte superior sombreada en gris, donde pone **help** presiona o pincha una vez en el botón izquierdo y verás las alternativas que dispone.

VER
>>ver % Muestra la versión de Matlab que se esta utilizando
WHAT
>>what %Da una lista de funciones .m en el directorio especificado.
Ejemplo: Imagina que tienes un directorio llamado "ejemplos" en el disquete lleno de funciones matlab y más cosas (documentos word, figuras etc...) y sólo quieres ver las funciones .m
>> what a:\ejemplos
LOOKFOR
>>lookfor % Acompañado de una palabra clave, busca una lista de funciones .m relacionadas con dicha palabra clave.
Ejemplo: si quieres conocer que funciones que realicen sumas tiene el matlab
>> lookfor sum
Verás como aparece un listado de comandos MATLAB junto con su explicación.

PWD
>>pwd % Indica el directorio en el que estoy trabajando
CD
1. >> cd %Te indica también (como pwd el directorio en el que te encuentras)
2. *>> cd ..* % se coloca en el directorio anterior.
3. >> cd a:\ejemplos % se coloca en el directorio a:/ejemplos

WHICH
>> which %Localiza la ubicación de una determinada función
Ejemplo:
>> which hilb %te dice donde se encuentra la función hilb
C:\MATLAB6p5\toolbox\matlab\elmat
DIR
>>dir %Da una lista del contenido del directorio en el que me encuentro

DESIGNACIÓN DE ESCALARES Y OPERACIONES

Designación de escalares

Si quieres dar un valor, por ejemplo 2, a un escalar llamado a, teclea:
>> a=2
Nota: la coma decimal en MATLAB se indica con un punto '.'
Ejemplo
>> a=2.34

Formato

En MATLAB existen diferentes modos o formatos de visualizar un resultado numérico. Dicho formato de salida se selecciona con el comando **format**. Usando la ayuda puedes conocer qué opciones de formato de salida existen.

Teclea
>> help format

Verás como para formato largo has de usar *format long*
MATLAB emplea por defecto el formato short.

Ejemplo:
>> format long
>> x=1.2
Observa cómo el MATLAB te devuelve el resultado con muchas cifras decimales

Operaciones con escalares

Suma : se emplea el símbolo +
Ejemplo: Suma a+b
>>a=4
>>b=2
>>s=a+b
Resta se emplea el símbolo -
Ejemplo:
>>r=a-b
Multiplicación se emplea el símbolo *
Ejemplo:
>>m=a*b
División a/b: se emplea el símbolo /
Ejemplo:
>>56/8
División b/a: se emplea el símbolo \
>>56\8
Potencia: se emplea el símbolo ^
Ejemplo, si a= 2, calcular a^2
>>a=2
>>a^2
Raíz cuadrada: se emplea el comando sqrt
Ejemplo: cálculo de la raiz cuadrada de 144
>>sqrt(144)

Funciones trigonométricas

Supón un determinado ángulo a, en radianes.
En la siguiente tabla te indicamos los comandos MATLAB empleados para definir las principales funciones trigonométricas, tanto directas como inversas.

Función	seno	coseno	tangente	seno-hiperbólico	coseno-hiperbólico	arco-seno	arco-coseno	arco-tangente
Comando MATLAB	b=sin(a)	c=cos(a)	d=tan(a)	sinh(a)	cosh(a)	asin(b)	acos(c)	atan(d)

NOTA: MATLAB **siempre** trabaja, por defecto, con **radianes**, por tanto primero tienes que pasar de grados a radianes (multiplicando por π y dividiendo por 180)
Ejemplo:
Calcula la tangente de 45 º
>>a=45 %defines el ángulo en grados
*>>a=a*pi/180 % lo pasas a radianes*
>>b=tan(a)

Otras funciones con escalares

exp: realiza la exponencial de un número
Ejemplo: e^3
>>exp(3)
log: realiza el logaritmo neperiano de un número
Ejemplo: ln(3)
>>log(3)
log10 : realiza el logaritmo en base 10 de un número
Ejemplo: log10(3)
>>log(3)
log2: realiza el logaritmo en base 2 de un número
Ejemplo: log2(3)
>>log2(3)
rem : te devuelve el resto de una división
Ejemplo: rem(a,b), te devuelve el resto de la división a/b
>>rem(1,3)
round:te redondea un número a su entero más próximo
Ejemplo: redondea 2.5
>>round(2.5)
sign: te devuelve el signo de un número. Con 1 positivo, y con –1 negativo.
Ejemplo:
>>sign(-3)

Visualización de variables creadas

Who : Hace un listado de las variables que se hayan creado en MATLAB
Ejemplo:
Crea una variable A=2
>> A=2
Crea una variable B=3
Visualiza si el MATLAB las tiene en memoria
>>who

Eliminación de variables creadas

Clear: borra una determinada variable creada.
Ejemplo: Si quieres borrar la variable A creada anteriormente

>>clear A
Si ahora haces who, verás como ya no esta
Clear all: borra todas las variables creadas
Ejemplo
>>clear all
>>who...verás como no te da ninguna respuesta. Esto es porque ya no tiene ninguna variable en memoria.
clc borra el contenido de la consola

DESIGNACIÓN DE MATRICES Y FUNCIONES MATRICIALES

Designación de matrices

Si quieres escribir por ejemplo la matriz

$$\begin{pmatrix} a_{11} & a_{12} & .....a_{1n} \\ .... & ..... & .... \\ a_{n1} & a_{n2} & a_{nn} \end{pmatrix}$$

Deberás indicar, **entre corchetes**, los valores de los elementos de cada fila separados por un espacio y distinguir cada fila de la siguiente por un **;**
>> A=[a12 a13 ... a1n; a21 a22 a23 ...a2n;....;an1 an2 an3 ....ann];
Ejemplo: escribir la matriz

$$\begin{pmatrix} 1 & 2 & 4 \\ 7 & 8 & 9 \\ 21 & 6 & 4 \end{pmatrix}$$

>> A=[1 2 4; 7 8 9; 21 6 4]

Denominación de los elementos de una matriz

Cada elemento de una matriz se designa con el nombre y, entre paréntesis, el número de la fila y de la columna donde se encuentra ese elemento

>>A(n,m) % con n el número de la fila y m el número de la columna
Ejemplo: en la matriz A anterior, el elemento de valor 21 se encuentra en la tercera fila de la primera columna, designándose entonces como:
>>A(3,1)

Elección de una fila o una columna

: quiere decir **todo** en MATLAB

a:b quiere decir "desde **a** hasta **b**"

Elección de una fila Para ello se seleccionan todas las columnas de una determinada fila

Elección de una columna: Para ello se seleccionan todas las filas de una determinada columna

Ejemplo:
Seleccionar la tercera columna de la matriz A anteriormente designada
>> A(:,3) % de este modo seleccionas todas las filas de la tercera columna
Seleccionar la segunda fila de la matriz A anteriormente designada
>> A(2,:) % de este modo seleccionas todas las columnas de la segunda fila
Selecciona la segunda y tercera fila de la primera columna de A
>>A(2:3,1)

Dimensión de una matriz

La dimensión de una matriz se indica con el comando **size**
>>[nf,nc]=size(A)
nf te indica el numero de filas
nc te indica el numero de columnas

Operaciones con matrices

Sean dos matrices A y B
Suma
>> A+B %suma elemento a elemento (las dimensiones de ambas matrices han de ser iguales)
Resta
>> A-B %resta elemento a elemento (las dimensiones de ambas matrices han de ser iguales)
Multiplicación
>>**A*B** %multiplicación matricial (el numero de columnas de A tiene que ser igual al numero de filas de B)
>> **A .*B** %multiplicación elemento a elemento (las dimensiones de ambas matrices han de ser iguales)
División
>> **A/B** %hace la división de A entre B , es decir A multiplicada por la inversa de B
>> **A\B** %hace la división de B entre A, es decir B multiplicada por la inversa de A
Potencia
>> **A.^b** %eleva cada uno de los elementos de A a un determinado numero b
Inversa
>> **inv(A)** %realiza la inversa de la matriz A

Otros[1]

>> **det(A)** %realiza el determinante de la matriz A
>> **A'** %realiza la matriz transpuesta de A
>> **diag (A)** %muestra los elementos de la diagonal de A

[1] Teclea help seguido de un determinado comando para saber más información sobre el mismo.
Si te interesa buscar algun otro comando que realice algo especifico teclea lookfor

>>**zeros(nf,nc)** %calcula una matriz de ceros con nf filas y nc columnas
>>**ones(nf,nc)** %calcula una matriz de ceros con nf filas y nc columnas
>>**eye(nf,nc)** %calcula una matriz de ceros con nf filas y nc columnas
>>**triu(A)** %toma la matriz triangular superior de A
>>**tril(A)** %toma la matriz triangular inferior de A
>> **randn(nf,nc)** %crea una matriz de números aleatorios de dimensiones nfxnc

CUESTIONARIO 1	
NOMBRE	
GRUPO	
ORDENADOR	

1. Haciendo uso de los comandos *help* o *helpwin* ejecuta y anota el resultado de ejecutar las siguientes sentencias:

>> helpwin help

>> helpwin cd

>> helpwin dir

>> helpwin who

2. Escribe el resultado de x=pi con el formato *long* y con el formato *short* y con el formato *long e*.

3. Dados a=6 y b=2. Indica la diferencia entre a/b y a\b

4. Indica el valor obtenido tras efectuar las siguientes operaciones en MATLAB. Muestra los comandos MATLAB empleados para llegar a la solución.

cos(30º)

sen(30º)

$\cos^2(30º)+\text{sen}^3(40º)$

5. Dadas las siguientes matrices: $A=\begin{pmatrix} 1 & 2 & 4 \\ 7 & 8 & 9 \\ 21 & 6 & 4 \end{pmatrix}$ *y* $B=\begin{pmatrix} 2 & 2 & 4 \\ 7 & 3 & 5 \\ 21 & 6 & 7 \end{pmatrix}$

Realiza las siguientes operaciones, indicando los comandos empleados:

Suma de las dos primeras columnas de A

Suma de las dos primeras filas de B

Cuadrado de la matriz A

Cuadrado de los elementos de la matriz B

Inversa de la matriz A

A*B

A.*B

A/B

B/A

6. Busca en Matlab, empleando el comando *lookfor*, una función interna que dé el valor máximo de cada uno de los elementos de las columnas de A.

MÓDULO 2

REPRESENTACIÓN GRÁFICA

Resumen de comandos relacionados con la representación gráfica:

COMANDO	OPERACIÓN
plot(x,y, 'r+:')	dibuja los vectores x,y de color rojo con el símbolo + y línea de puntos. Puedes cambiar : COLOR: b blue, g green, r red, c cyan SÍMBOLO: . point, o circle, x x-mark, + plus LÍNEAS: - solid, : dotted, -. dashdot, -- dashed
plot(x,y)	dibuja dos vectores por defecto en azul y con línea continua.
plot(x,y,'r+:',x,z,'b*:')	dibuja dos graficas una en rojo con + y otra en azul con *
plot(A(2,:))	dibuja la fila 2 de una matriz A
figure	para dibujar varias figuras
hold on	para mantener la grafica anterior dentro de una misma figura
hold off	Deja de mantener la grafica anterior en el momento en el que se dibuja otra.
xlim([100 200])	limita el eje X a los puntos entre 100 y 200
ylim([100 200])	limita el eje Y a los puntos entre 100 y 200
xlabel('eje x')	etiqueta al eje X
ylabel('eje y')	etiqueta al eje Y
title('Titulo')	añade un texto encima de la grafica
text(x,y,'texto')	añade un texto en la posición relativa x,y
legend('n=2','n=4')	dibuja una leyenda
grid on	añade un grid
subplot(325), plot(x,y)	dibuja en una matriz de 3x2 gráficas, en la 5ª posición
stem(x,y)	dibuja en barrotes
bar(x,y)	dibuja en diagrama de barras
stairs(x,y)	dibuja en escalera
feather(x,y)	dibuja con flechas
semilogx(x,y)	dibuja con el eje x en escala log10
loglog(x,y)	dibuja con el eje x y el y en escala log10

CUESTIONARIO 2			
NOMBRE		**FECHA**	
GRUPO		**ORDENADOR**	

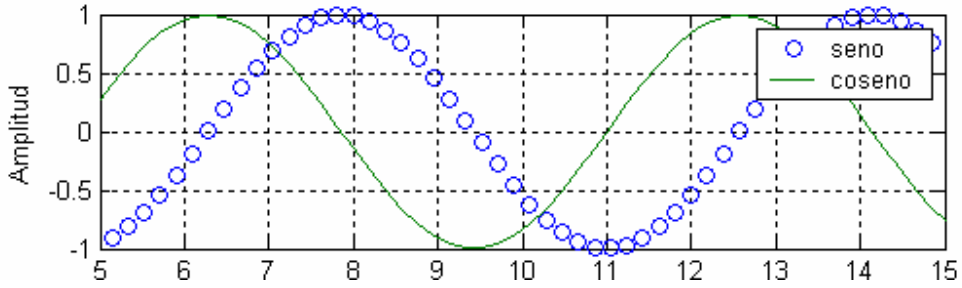
1. Haciendo uso del comando LINSPACE crea una variable x que vaya desde 0 a 6π con 100 valores.

2. Crea dos variables "y" y "z" que sea el sean el seno y el coseno de x.

3. Indica las sentencias que tienes que escribir para realizar una grafica que represente "y" frente a "x" en la que aparezcan como etiquetas del eje "X" "Nº de muestras" y en el eje "Y" "Seno".

4. Indica las sentencias que tienes que escribir para realizar una grafica que represente z frente a "X" en el que aparezca como etiquetas del eje "X" "Nº de muestras" y en el eje "Y" "Coseno". La grafica no debe de tener línea y los símbolos de los puntos tienen que ser rojos y redondos.

5. Indica las sentencias que tienes que escribir para realizar una grafica que sea como la de la figura:



6. Indica las sentencias que tienes que escribir para representar en una misma gráfica las figuras de los ejercicios 3,4 y 5 , una debajo de la otra y que tenga por titulo general 'Representación grafica'. Utiliza el comando subplot.

CREACIÓN DE UN FICHERO *.m

DEFINICIÓN:

Un fichero *.m es un fichero con un conjunto de comandos que Matlab puede interpretar. Distinguiremos dos tipos de ficheros *.m dentro del Matlab:

- **Fichero script** : es un fichero 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.

- **Función:** es un fichero con un conjunto de comandos que necesita una variable de entrada para que Matlab pueda ejecutar. Puede tener variables de salida. Todos los fichero *.m que sean funciones tiene que empezar por :

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

  siendo entrada1, entrada2 ,..... parámetros necesarios para la función y salida1, salida2, ... parámetros opcionales de salida.

CREACIÓN DE COMENTARIOS EN UN FICHERO .M

Para escribir un comentario dentro de un fichero *.m de manera que el compilador de Matlab no lo interprete se escribirá el símbolo % delante de todas las sentencias que no se quiere que se ejecuten.

CREACIÓN DE AYUDA EN UN FICHERO .M

Si ponemos nada más empezar el fichero una serie de comentarios que indiquen los procesos que realizan la función, así como un ejemplo de cómo corre el programa nos aparecerá esta información al escribir desde la consola “help nombrefichero.m”

CREACIÓN DE UN FICHERO .M

1. Pinchar File en el menú de ventanas de MATLAB.
2. Seleccionar la opción New y M-file (con ello entraremos en el editor de MATLAB).
3. Escribe tu script

Ejemplo:

```
% Creación de una figura conteniendo las funciones seno y coseno.
x=linspace(0,2*pi,30);
y=sin(x);
plot(x,y,x,y,'*',x,cos(x),x,cos(x),'+')
grid
xlabel('Variable independiente')
ylabel('Variables dependientes')
title('Primer ejemplo de gráficas')
```

GRABACIÓN DE UN FICHERO .M

1. Vete a la ventana File y selecciona Save as.
2. Graba el nombre de este fichero en el disco A

   Ejemplo:
   En el ejemplo anterior, graba tu script con el nombre grafica.m

EJECUCIÓN DE FUNCIONES Y SCRIPTS

- Para ejecutar un script desde la línea de comandos de MATLAB tienes que poner el nombre de dicho fichero (sin extensión m)
  Ejemplo: ejecuta el programa grafica
  >> grafica
  Escribe el resultado o resultados de la ejecución.

- Para ejecutar una función desde la línea de comandos de MATLAB tienes que poner el nombre de dicho fichero (en minúsculas) y, entre paréntesis las variables de entrada.
  Ejemplo: [salida1, salida2,...] =nombre(entrada1, entrada2,...);

IMPORTANTE:

- En el caso de funciones, el nombre del fichero tiene que ser el mismo que aparece en la línea *function*.

- La extensión .m es, tanto para script como para funciones, obligatoria.

- Si indicas la ayuda dentro del script y quieres verla desde la línea de comandos, ejecuta:

  >> help “nombre del fichero”

- Si quieres que, desde la línea de comandos, aparezca el contenido completo del fichero .m, ejecuta

>> type “nombre del fichero”

Ejemplo:
¿Qué hace el programa grafica.m que has editado?
Para ello teclea:
>>help grafica
Muestra el contenido del programa grafica.m
>>type grafica.m

EJERCICIO PRÁCTICO 1:

(a) Partiendo del programa ***grafica.m***, realiza las siguientes modificaciones:

- Coloca una primera sentencia como ***function grafica2(x,y)***
- Elimina las sentencias en las que se define X e Y.
- Graba este nuevo programa con el nombre de ***grafica2.m*** (Recuerda que debes hacerlo con Save as

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

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

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

EJERCICIO PRÁCTICO 2:

(a) Crea 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);
```

(b) Busca el significado de las órdenes que no entiendas.

(c) Ejecuta las sentencias:

```
>>y=[1 2 3 4 3 2 1]
>>[media,desviacion]=medesv(y)
```

Escribe los resultados.

¿por qué crees que, si en el fichero se llama medesv(x), desde la barra de comandos funciona igual poniendo medesv(y)?

(d) Vuelve a preguntar por el valor de las magnitudes anteriores, recuerda que se hace ejecutando:

```
>>media
>>desviacion
```

Estas instrucciones nos indican que hemos almacenado los valores de la media y la desviación en las variables MEDIA y DESVIACION, lo cual nos permitirá poderlas usar más adelante. El comando who, que vimos en el modulo anterior, también te indica que has almacenado dichas variables.

CUESTIONARIO 2 (continuación)			
NOMBRE		**FECHA**	
GRUPO		**ORDENADOR**	

Nota : Todos los fichero que haremos en clase tienen que llevar un encabezamiento con el nombre y apellidos del alumnos y en número del PC, así como una breve descripción de lo que realiza el programa.

1. Crea un script que se llame script1.m y que realice la grafica del problema 5 del cuestionario.

2. Modifica el script anterior para convertirlo en una función (funcion1.m) en la cual los límites del eje x que representa sean dos parámetros que introducimos desde el teclado.

3. Modifica la función anterior para que me de cómo parámetros de salida dos variables con los valores del seno y del coseno. (funcion2.m)

MÓDULO 3

OPERACIONES E INSTRUCCIONES RESERVADAS EN MATLAB

A continuación tienes un resumen de algunas de las instrucciones básicas de Matlab.

CONSTRUCCIONES DE CONTROL

CONSTRUCCIÓN	**ESTRUCTURA**
BUCLES **repite una orden dada por una(s)** *sentencia (s)* **un número de veces determinado por una** *variable*	**for** *variable= expresion* *sentencia* **end**
IF **Ejecuta un conjunto de** *sentencias* **si una** *condición* **se cumple**	**if** *condicion* *sentencias* **end**

IF CON ANIDAMIENTO
 	
Ejecuta un conjunto de *sentencias* **si una** *condición1* **se cumple, otro conjunto de** *sentencias* **si la** *condicion2* **se cumple (usando elseif) y, si ninguna de estas** *condiciones* **se cumplen(con else), ejecuta otra** *sentencia*
(<u>nota</u>:se pueden usar tantos elseif como condiciones quieras indicar**)**
	if *condicion1*
	  sentencias
	elseif *condicion2*
	  sentencias
	else
	  sentencias
	..
	end

WHILE **Repite un conjunto de** *sentencias* **mientras que se cumple la** *condición*	**while** *condición* *sentencia* **end**

CONSTRUCCIÓN DE CONDICIONES PARA SENTENCIAS IF Y WHILE

Para construir condiciones se emplean una serie de operadores, llamados operadores lógicos y operadores de comparación, que se indican en la siguiente tabla.

OPERADORES ÚTILES	
OPERADORES LÓGICOS	**OPERADORES DE COMPARACIÓN**
& AND	**<** Menor que **< =** Menor o igual que
| OR	**>** Mayor que **> =** Mayor o igual que
~ NOT	**= =** Igual que **~ =** Distinto (~: presiona ATL 126)

ENTRADA Y SALIDA DE TEXTO

disp......visualiza texto en pantalla
ejemplo: **disp**(**'**hola**'**)

error.....visualiza texto en caso de error y el
ejemplo: *error****('no se puede ejecutar')*** ***termina el fichero .m.***

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

Ejemplos

BUCLES

Ejemplo sencillo

Creación de un vector x de 10 elementos, cuyas componentes indican el valor del seno de un ángulo cuando que varía de $\pi/10$ a π.

```
for n=1:10
    x(n)=sin(n*pi/10);
    disp(n)
    disp(x)
    pause     % Hace una parada después  de cada paso del bucle
end
```

Fíjate que al ejecutarlo tienes por pantalla el siguiente resultado

```
   1
   0.3090
    2
   0.3090    0.5878
    3
   0.3090    0.5878    0.8090
    4
   0.3090    0.5878    0.8090    0.9511
    5
   0.3090    0.5878    0.8090    0.9511    1.0000
    6
   0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
    7
   0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
    8
   0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090    0.5878
    9
   0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090    0.5878    0.3090
   10
   0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090    0.5878    0.3090    0.0000
```

Esto te indica como, para cada n, se van creando cada uno de los elementos del vector hasta completar el bucle, momento en el cual estarán calculados todos los elementos de x.

Ejemplo de bucle con anidamiento

Creación de una matriz de 5 filas por 5 columnas en la que los elementos sean la suma de la posición de la fila y la columna correspondiente. Por pantalla ha de aparecer la fila en la que se encuentra en cada momento.

```
for n=1:5
        for m=5:-1:1
                a(n,m)=n+m;
end
disp(n)
disp(a)
pause   % hace una parada después de cada fila
end
```

Observa el programa anterior. ¿Por qué aparece por pantalla esta solución?

```
1
    2    3    4    5    6
```

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

IF

Ejemplo 1 sencillo

Dado un valor de m, calcula el cuadrado de ese número sólo si m es mayor que 5. En caso contrario escribe un mensaje de error.

```
function ejemif1(valor)
% función que calcula el cuadrado de un número si es mayor que 5
if valor>=5
    c=valor^2;
     disp(c)
    disp('El numero es mayor que 5')
else
    disp('El numero es menor que 5')
    error('Introduce un numero mayor que 5')
end
```

Ejemplo 2 de if con anidamiento

En el ejemplo anterior podemos poner otras sentencias para los casos de M menor o igual que 5, en lugar de poner un mensaje de error.

```
function ejemif2(valor)
% función que calcula el cuadrado de un número si es mayor que 5 el cubo del mismo si es
mayor que 10 y da un error en caso contrario
if valor>=5
    c=valor^2
    disp('El numero es mayor que 5')
     if valor>10
          c=valor^3
         disp('El numero es mayor que 10')
      end
else
    disp('El numero es menor que 5')
    error('Introduce un numero mayor que 5')
end
```

Ejemplo 3 de if con anidamiento

Crea una función llamada *suma* que tenga como valores de entrada dos números (a y b) de modo que calcule la suma en valor absoluto de dos números cualesquiera**.**

```
function [c]=suma(a,b)
% Función que realiza la suma en valor absoluto de dos variables
if a >0 & b >0
   c=a+b;
elseif a >0 & b <0
   c=a-b;
elseif  a <0 & b >0
   c=-a+b
elseif a<0 & b<0
   c=-a-b
end
```

WHILE

Ejemplo 4. Crea un script que calcule la suma de los números enteros empezando en el 1, hasta que la suma exceda de 100.

Solución:

```
suma=0;n=1;
while (suma)<=100
   suma=suma+n;
   n=n+1;
   disp(suma)
end
```

¿Por qué obtienes esta salida?

```
 1
 3
 6
10
15
21
28
36
45
55
66
78
91
105
```

REALIZA LOS SIGUIENTES PROGRAMAS

NO OLVIDES GUARDARLOS EN TU DISQUETE CON EL NOMBRE QUE SE TE INDICA A CONTINUACIÓN

1. Crea dos programas **.m** que contengan los dos ejemplos que se indican dentro de la sección de ejemplos de **BUCLES**. Estos programas se llamarán:

ejemfor1.m (contendrá el conjunto de sentencias del ejemplo sencillo de **for**).
ejemfor2.m (contendrá el conjunto de sentencias del ejemplo más complejo de **for**).

2. Crea tres programas **.m** que contengan los tres ejemplos que se indican dentro de la sección de ejemplos **IF** anterior. Estos programas se llamarán:
ejemif1.m (contendrá el conjunto de sentencias del ejemplo 1).
ejemif2.m (contendrá el conjunto de sentencias del ejemplo 2)
ejemif3.m (contendrá el conjunto de sentencias del ejemplo 3)

Explica los resultados que obtienes, indicando si son lógicos, en el ejemplo 2 para los siguientes valores de m:
(a) –3.0
(b) 0
(c) 1.3
(d) 1
(e) 5

3. Crea un programa **.m** que contengan el ejemplo que se indica dentro de la sección WHILE. Este programa se llamará **ejemwhile.m** (contendrá el conjunto de sentencias del ejemplo de **while**).

CUESTIONARIO 3			
NOMBRE		**FECHA**	
GRUPO		**ORDENADOR**	

Nota : Todos los fichero que haremos en clase tienen que llevar un encabezamiento con el nombre y apellidos del alumnos y en número del PC, así como una breve descripción de lo que realiza el programa.

1. Crea una función **C=traspo(A)** en MATLAB para obtener la matriz **C** como traspuesta de la matriz **A**, sólo en el caso en que ésta sea una matriz cuadrada. En caso contrario, el programa debe mostrar un mensaje en pantalla. **Utilícense bucles y sentencias *if y for*.**

Nota: Dada una matriz A, matlab calcula su traspuesta AT como A'. En este ejercicio podrás emplear sólo A' para comprobar tu resultado

Guarda esta función en tu disquete como **traspo.m**

```
function B= traspo(A)
%clear all; close all;
[m n]=size(A);
if m~=n, error('La matriz no es cuadrada'), end
for i=1:m
   for j=1:n
      B(i,j)=A(j,i);
   end
end
```

2. Crea una función **A=matrix(n)** en MATLAB que asigne valores a una matriz cuadrada de dimensión n (n filas y n columnas), en que:

a. El elemento A(i,j)=8j-5i si i>j
b. los elementos de la diagonal son la unidad.
c. el resto son cero.

Utilícense bucles y sentencias *if y for*. Aplíquese para n=2 y n=3.

Guarda esta función en tu disquete como **matrix.m**

```
function A=matrix(n)
for i=1:n
   for j=1:n
      if i>j , A(i,j)=8*j-5*i; ,end
      if i==j, A(i,j)=1; ,end
      if i<j, A(i,j)=0; , end
   end
end
```

```
A =
     1     0     0     0     0     0
    -2     1     0     0     0     0
    -7     1     1     0     0     0
   -12    -4     4     1     0     0
   -17    -9    -1     7     1     0
   -22   -14    -6     2    10     1
```

CUESTIONARIO 3 (continuación)			
NOMBRE		**FECHA**	
GRUPO		**ORDENADOR**	

3. Crea una función **M=factor(n)** en MATLAB que determine el factorial de un número natural **n**. La ejecución del programa debe incluir un conjunto de sentencias tal que:
   a. en el caso de que **n** no sea positivo evite la ejecución del programa, dando un mensaje explicativo
   b. el resultado final ha de aparecer en pantalla con un mensaje

   Aplica la función **factor** al caso de n=8.

Guarda esta función en tu disquete como **factor.m**

4. Crea una función llamada **serie** en MATLAB tal que sus datos de entrada sean:
El **primer término** de una serie aritmética (**a_1**)
Un **valor positivo** que indique la diferencia entre dos términos consecutivos de la serie aritmética (**d**)
Una cota positiva (**cota**) y los datos de salida sean:
El número de términos **n** cuya suma **S** supere minimamente el valor dado por la **cota** La suma **S**

La ejecución del programa debe incluir:
   a. Un conjunto de sentencias que, en el caso de que **cota** no sea positiva, evite la ejecución del programa, dando un mensaje explicativo.
   b. La aparición en pantalla de los resultados parciales de **n** y **S** .

Aplica la función **serie** al caso de a1= -1, d=3 y cota=250.

Guarda esta función en tu disquete como **serie.m**

MÓDULO 4 - REPASO

Instrucciones genéricas para un programa.

1. Al principio de cada programa pondremos el nombre, la fecha y el número del PC en el que se trabaja. A continuación, se escribirá una **descripción de lo que realiza el programa** junto con una sentencia que indique lo que tendremos que teclear, de forma genérica desde la consola del Matlab, para que se ejecute, así como un ejemplo que funcione.
   De esta manera, cuando se teclea desde el Matlab "help" y el nombre del programa, aparecerá toda esta información.

   Ejemplo:

```
% Nombre : Aniseto Rodríguez
% 14-II-04
% Numero Pc : 23
% Funcion que suma todos los numeros  pares entre dos numeros pares % que
introducimos parámetros de entrada
% Si alguno de los numeros no son pares da un error y sale del programa
% El programa devuelve el valor de la suma
% Teclear :  sumapares(a,b)
% Ejemplo : sumapares (10,20)
function salida=sumapares(vi,vf)
```

2. Explicación de las variables de entrada

```
% Defino las variables para que las entienda
% vi valor inicial a partir del cual empiezo a sumar
% vf valor final hasta el que tengo que sumar
```

3. Comprobación de los valores de entrada

```
% Comprobacion de valores de entrada
a=rem(vi,2);  % la funcion rem da el resto de la division
b=rem(vf,2);
if a~=0
    disp('El primer numero introducido no es par')
    return   % salgo del programa
end
if b~=0
    disp('El segundo numero introducido no es par')
    return
end
if vf<vi
    disp('El segundo numero tiene que ser mayor que el primero')
    return
end
```

4. Inicialización de las variables auxiliares

```
% Inicializa variables dandole un nombre que nos diga algo
sum=0;
cont=0;
```

5. Ejecución del programa

```
% Ejecuto el programa
fprintf('\n Contador  Valor      Suma')
for valor=vi:2:vf  % Va desde vi hasta vf de dos en dos
   cont=cont+1;
   sum=sum+valor;
   fprintf('\n     %2i      %3i     %5i', cont, valor, sum)
   % Saco por pantalla los calculos
end
salida=sum;

%FIN DEL PROGRAMA
```

6. Fin del programa

Antes de ejecutar cualquier programa comprobamos el directorio de trabajo con “pwd” y que el programa que queremos ejecutar esta en ese directorio “dir”. Es conveniente borrar todas variables que tenga el entorno mediante la sentencia “clear all” , borramos también la pantalla con “clc” y a continuación ejecutamos dicho programa.

CUESTIONARIO 4			
NOMBRE		**FECHA**	
GRUPO		**ORDENADOR**	

Nota : Todos los fichero que haremos en clase tienen que llevar un encabezamiento con el nombre y apellidos del alumno y eL número del PC, así como una breve descripción de lo que realiza el programa y un ejemplo de lo que habría que teclear.

1 – Sabiendo que las coordenadas cartesianas de una circunferencia son de la forma x=r cos(θ) , y=r sen(θ) 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.

```
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')
```

2 – Igual que el anterior solo que, 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.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

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

_____
_____
_____
_____
_____

4 – 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

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

MÓDULO 5

ARQUITECTURA DEL COMPUTADOR. SISTEMAS DE ECUACIONES LINEALES

A. ARITMÉTICA DEL COMPUTADOR

CUESTIONARIO 5.1			
NOMBRE		**FECHA**	
GRUPO		**ORDENADOR**	

Nota : Todos los programas que hagamos en clase tienen que llevar un encabezamiento con el nombre y apellidos del alumno y eL número del PC, así como una breve descripción de lo que realiza el programa y un ejemplo de lo que habría que teclear.

Representación en máquina de números enteros sin signo :

1. Realiza un programa que se llame int2bin.m que represente un número entero sin signo en el sistema binario. (opcional: incluye la representación binaria de números enteros con signo)
Para pasa un número de base 10 a base 2 seguiremos los paso siguientes

Pasos:
Preguntar si el número es entero
i=1
Dividir por 2 el número obteniendo un cociente y un resto
El bit de orden i es igual al resto de la división
i=i+1
Volver al 1^{er} paso dividiendo el nuevo cociente por 2. Realizar este bucle mientras que el cociente sea >=2
El primer bit es el último cociente

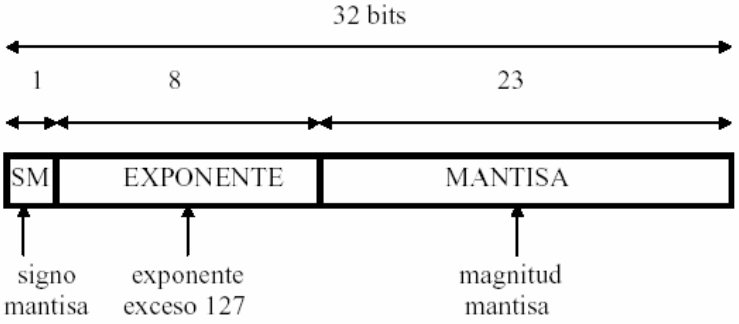
2. Representación de números reales: Recuerda que, dado un número real menor que 1 en base 10 $(r)_{10}$, su representación en el sistema binario puede obtenerse mediante la secuencia de operaciones siguiente:

Pasos:
Preguntar si el número esta entre 0 y 1
i=1
Multiplicar por 2 el número
Si es mayor que 1, entonces el bit de orden i es igual a 1 y restamos 1 al número
Si NO es mayor que 1, entonces el bit a_i es igual a 0
i=i+1
Volver al paso 3 hasta que obtengamos los bits deseados o hasta terminar el proceso (que el numero sea 0).

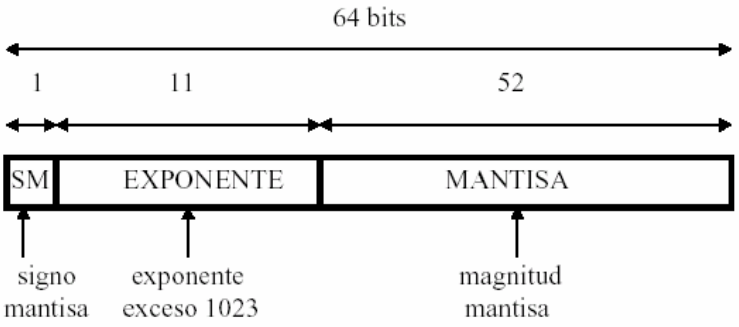
Realiza un programa que se llame flo2bin.m que represente un número real en coma decimal flotante para valores entre 0 y 1 en el sistema binario. El programa tiene que limitar el nº de bit Aplícalo al caso de x=0.7 y x=0.5.

3 El Estándar IEE754.

Según el Estándar IEE754, la mantisa está representada como un número binario con signo y el exponente en exceso 127, lo cual quiere decir que el ordenador no podrá emplear un exponente en binario fuera del rango [-127,127]. Esto quiere decir que el computador no podrá manejar números con una magnitud mas grande $2^{127} = 10^{38}$ ni más pequeña que $2^{-127}=10^{-38}$



Exponente	Mantisa	Representa
-127 (combinación binaria 0)	0	± 0
-127 (combinación binaria 0)	$\neq 0$	$(-1)^{sm} \cdot (0.m_{23}...m_0) \cdot 2^{-126}$
[-126, 127] (combinación binaria [1, 254])	-	$(-1)^{sm} \cdot (1.m_{23}...m_0) \cdot 2^{(e_7...e_0)_2-127}$
128 (combinación binaria 255)	0	$\pm$inf
128 (combinación binaria 255)	$\neq 0$	NaN



Exponente	Mantisa	Representa
-1023 (combinación binaria 0)	0	± 0
-1023 (combinación binaria 0)	$\neq 0$	$(-1)^{sm} \cdot (0.m_{51}...m_0) \cdot 2^{-1022}$
[-1022,1023] (combinación binaria [1, 2047])	-	$(-1)^{sm} \cdot (1.m_{51}...m_0) \cdot 2^{(e_{10}...e_0)_2-1023}$
1024 (combinación binaria 2048)	0	$\pm$inf
1024 (combinación binaria 2048)	$\neq 0$	NaN

4 - Comprueba, usando MATLAB, si el computador donde realizamos las prácticas sigue el estándar IEEE 754 para representar los números reales. Recordar que Matlab utiliza doble precisión. Escribe un script llamado precision.m que, para conocer la precisión del computador, realice en cada iteración de un bucle la suma $1.0+2^{-k}$ e incremente k en una unidad, de modo que cuando la suma sea igual a 1.0 el bucle pare. El hecho de que ambas cantidades sean iguales significa que hemos excedido la precisión del computador. Entonces k-1 es el número de bits de la mantisa y 2^{-k+1} es la precisión, épsilon o error de redondeo unitario del computador.

B. RESOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES-
Métodos Directos

CUESTIONARIO 5.2			
NOMBRE		**FECHA**	
GRUPO		**ORDENADOR**	

Nota : Todos los programas que hagamos en clase tienen que llevar un encabezamiento con el nombre y apellidos del alumno y eL número del PC, así como una breve descripción de lo que realiza el programa y un ejemplo de lo que habría que teclear.

5 Sistema Triangular Inferior. Método de sustitución progresiva

Sea el siguiente sistema de ecuaciones escrito en forma matricial:

$$\begin{pmatrix} a_{11}\ 0\ 0\ 0.....0 \\ a_{21}\ a_{22}\ 0\ 0\ 0.....0 \\ . \\ . \\ a_{n1}\ a_{n2}\ .....a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ .. \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ .. \\ b_n \end{pmatrix}$$

$k = 1$	$x_1 = \dfrac{b_1}{a_{11}}$
$k \neq 1$	$x_k = \dfrac{\left[b_k - \sum_{i=1}^{k-1} a_{ki}\ x_i \right]}{a_{kk}}$

Escribe una función MATLAB sp.m que admita como parámetros de entrada una matriz triangular inferior *A* y un vector columna *b*, con la parte derecha del sistema de ecuaciones, y devuelva, como parámetro de salida *x*, la solución del sistema obtenida aplicando el algoritmo de sustitución progresiva. Aplicar este algoritmo a: :

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

6 Sistema Triangular Superior. Método de sustitución regresiva

Siguiendo el razonamiento equivalente al ejercicio 5, Escribe una función MATLAB llamada sr.m que admita como parámetros de entrada una matriz triangular superior *A* y un vector columna *b* con la parte derecha del sistema de ecuaciones y devuelva, como parámetro de salida *x*, la solución del sistema.

Aplicar este algoritmo para resolver el sistema de ecuaciones:

$$\begin{bmatrix} 1 & 4 & 5 \\ 0 & 3 & 2 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 6 \end{bmatrix}$$

7. Método de Eliminación Gaussiana. Reducción Progresiva.

El algoritmo de eliminación gaussiana está formado por una secuencia de operaciones que transforma el sistema en **triangular superior(TS)**, mediante la eliminación de los n-1 de la ultima fila, a continuación los n-2 de la penúltima y así sucesivamente. Una vez convertida la matriz de los coeficientes original en TS El sistema se resuelve empleando el método de sustitución regresiva.

Escribe una función **rp.m** que admita como parámetros de entrada una matriz *A* y un vector columna *b* con la parte derecha del sistema de ecuaciones y devuelva, como parámetro de salida *x*, la matriz *A* y el vector *b*, tras haber aplicado el algoritmo de reducción progresiva.
Resolver el sistema de ecuaciones *Ax=b.* Aplicar este algoritmo sobre el siguiente sistema de ecuaciones:

$$\begin{bmatrix} 3 & -1 & 2 \\ 1 & 2 & 3 \\ 2 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 12 \\ 11 \\ 2 \end{bmatrix}$$

8. Descomposición de Cholesky:

Escribir un programa llamado **cholesky.m** que realice la descomposición de Cholesky de una matriz. Para comprobar que es correcta se puede comparar con la función de MATLAB chol

9. Hay casos en los que la solución numérica obtenida en un sistema de Ecuaciones Lineales no es fiable. Estudiar el siguiente sistema:

0.832 *x*1 + 0.448 *x*2 = 1
0.784 *x*1 + 0.421 *x*2 = 0

Ahora resolver el mismo sistema pero con el coeficiente *a22* incrementado en 0.001:
¿Qué ha ocurrido? ¿Por qué?

MÓDULO 6

RESOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES MÉTODOS ITERATIVOS

OJO!!! TODOS LOS PROGRAMAS DEBEN DE LLEVAR ANTES LA CONDICIÓN DE QUE LA MATRIZ A SEA CUADRADA, CON DETERMINANTE DISTINTO DE CERO Y BIEN CONDICIONADA.

1 - Escribe una función llamada **x1=jacobi1(A,b,x0)** que admita como parámetros de entrada la matriz del sistema de ecuaciones *A*, un vector columna *b* con la parte derecha del sistema de ecuaciones, y una aproximación a la solución *x0* y devuelva, como parámetro de salida, el resultado, de aplicar **una iteración** de Jacobi *xn*. Implementar la función con que utilice exclusivamente operaciones matriciales, $x1=D^{-1} * [(b-(L+U) * x0]$ donde dividimos la matriz A en parte diagonal (D), parte estrictamente inferior (L) y parte estrictamente superior (U) A=D+L+U.

2 -Escribe una función llamada **x1=jacobi2(A,b,x0)** similar al anterior pero con el uso de bucles for. (.... X1(i)=xn(i)-A(i,j)*x0(j) ....) (Nota: Ver hoja de ayuda para la programación del método de jacobi al final del capítulo)

3 - Escribe una función llamada **xn = mjacobi(A,b,x0,tolerancia)** que admita como parámetros de entrada una matriz *A*, un vector columna *b* con la parte derecha del sistema de ecuaciones, una aproximación inicial a la solución *x0* y un valor de *tolerancia* y devuelva, como parámetro de salida, la solución *x* del sistema de ecuaciones dentro de la tolerancia dada. Además, debe visualizar el error y la solución aproximada en **cada iteración**. La función deberá llamar a la función jacobi (1 o 2). Realizar dentro de la misma función realizar la representación grafica de la norma de x-x0 frente al número de iteraciones.
Usar esta función para obtener la solución del siguiente sistema de ecuaciones dentro de un error 0.001 ¿Cuántas iteraciones son necesarias? ¿Y para un error de 0.00001? Comprobar el resultado con: **x** = **A**\ **b**

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

4 - a) Escribe una función llamada **xs1 = gseidel(A,b,xs)** que realice una iteración de Gauss-Seidel sobre el sistema de ecuaciones introducido como argumento de entrada.
b) Implementar la misma función manejando matrices directamente.

5 - Escribe una función llamada **x = mgseidel(A,b,x0,tolerancia)** que admita como parámetros de entrada una matriz *A*, un vector columna *b* con la parte derecha del sistema de ecuaciones, una aproximación inicial a la solución final *x0*, y un valor de *tolerancia* y devuelva, como parámetro de salida *x*, la solución del sistema de ecuaciones dentro de la tolerancia dada utilizando el método de Gauss-Seidel. Además, debe visualizar el error y la solución aproximada en cada iteración.

6 - a) Escribe una función llamada **xs1=sor(A,b,xs,w)** que admita como parámetros de entrada la matriz del sistema de ecuaciones *A*, un vector columna *b* con la parte derecha del sistema de ecuaciones, una aproximación a la solución *xs* y un peso *w,* y devuelva, como parámetro de salida *xs1*, el resultado de aplicar una SOR con el peso dado.
b) Implementar la misma función utilizando las expresiones matriciales
7 - Escribe una función llamada **x=msor(A,b,x0,tolerancia,w)** que admita como parámetros de entrada la matriz del sistema *A*, un vector columna *b* con la parte derecha del sistema de ecuaciones, una aproximación inicial *x0* a la solución final, un valor de *tolerancia* y un peso *w*, y devuelva, como parámetro de salida *x*, la solución del sistema de ecuaciones dentro de la tolerancia dada utilizando el método SOR. Además, debe visualizar el error y la solución aproximada en cada iteración.

8 - Supongamos el siguiente sistema de ecuaciones:

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

Si queremos alcanzar la solución con una tolerancia igual a 0.00001 y con una aproximación inicial a la solución (0 0 0 0). Haz las siguientes pruebas:
- ¿Cuantas iteraciones de **Jacobi** hacen falta para alcanzar la solución?
- ¿Cuantas iteraciones de **Gauss-Seidel** hacen falta para alcanzar la solución?
- Variando el peso del método **SOR** desde 1.0 a 1.9 con un incremento 0.1,
  ¿Cuantas iteraciones hacen falta para cada peso y qué peso minimiza el
  número de iteraciones?

9 - Escribe una función llamada **xs1=jacobiw(A,b,xs,w)** que admita como parámetros de entrada la matriz del sistema de ecuaciones *A*, un vector columna *b* con la parte derecha del sistema de ecuaciones, una aproximación *xs* a la solución y un peso *w,* y devuelva, como parámetro de salida, el resultado *xs1* de aplicar una iteración de Jacobi amortiguado con el peso dado.

10 - Escribe una función llamada **x=mjacobiw(A,b,x0,w,tolerancia)** que admita como parámetros de entrada la matriz del sistema *A*, un vector columna *b* con la parte derecha del sistema de ecuaciones, una aproximación inicial a la solución *x0*, el peso *w* y un valor de *tolerancia* y devuelva, como parámetro de salida, la solución *x* del sistema de ecuaciones dentro de la tolerancia dada. Además, debe visualizar el error y la solución aproximada en cada iteración.
a) Usar esta función para obtener la solución del siguiente sistema de ecuaciones dentro de un error 0.001 utilizando un peso de 0.5 ¿Cuántas iteraciones son necesarias?
b) Aplicar la función jacobiw.m al sistema de ecuaciones del ejercicio 3, considerando una tolerancia de 0.001 y un peso elegido por el alumno tal que el proceso converja más rápidamente respecto al *método de Jacobi*.

AYUDA PARA LA PROGRAMACIÓN DEL MÉTODO DE JACOBI

A. CON UNA SOLA ITERACIÓN

Crearemos una función llamada **jacobi1.m.** Los pasos a seguir son los siguientes:

1.definir la función Jacobi con las siguientes características

- entradas: :
  - i. matriz de coeficientes **A**
  - ii.matriz **b**
  - iii. solucion inicial
- salidas
  - i.solución final **x**

2.Definir la dimensión de la matriz A

3.Comenzar bucle en i para calcular cada una de las n (dimensión de la matriz) componentes de la solución.

4.Determinar un primer valor para cada componente

$x_i=b_i$

5.crear un nuevo bucle en j, para valores de j distintos a i

6.Añadir a la solución xi el resto de los términos del sumatorio

$x_i=x_i-a_{ij}\ x0_j$

7. Salirse del bucle en j, y dividir la solución por a_{ii}

$x_i=x_i/a_{ii}$

8. salirse del bucle en i

B. CON MAS DE UNA SOLA ITERACIÓN

Crearemos una función llamada **mjacobi.m.**

1.definir la función jacobi con las siguientes características

- entradas: :
  - i. matriz de coeficientes **A**
  - ii.matriz **b**
  - iii.solución inicial $\mathbf{x_o}$
  - iv.valor de tolerancia **tol** (se suele tomar 0.01)
- salidas
  - i.solución final **x**
  - ii.numero de iteraciones **it**
  - iii.valor que va tomando la cota ($|x-x_o|$) en cada iteración

Definir la dimensión de la matriz A

2. Inicializar : **it**, **cota** y un primer valor de la solución **x** (lo hallaremos a partir de jacobi1)

```
it=1
x=jacobi1(A,b, xo);
cota=abs(x-xo);
```

comenzar un while en el que la condición sea que "mientras la cota sea mayor que la tolerancia que vuelva a aplicar jacobi1 :utilizando la iteración anterior". Dentro del while poner un fprintf con (numero de iteración, valor de la solución x, valor de la cota)

3. Representar el valor de la cota frente al número de iteraciones

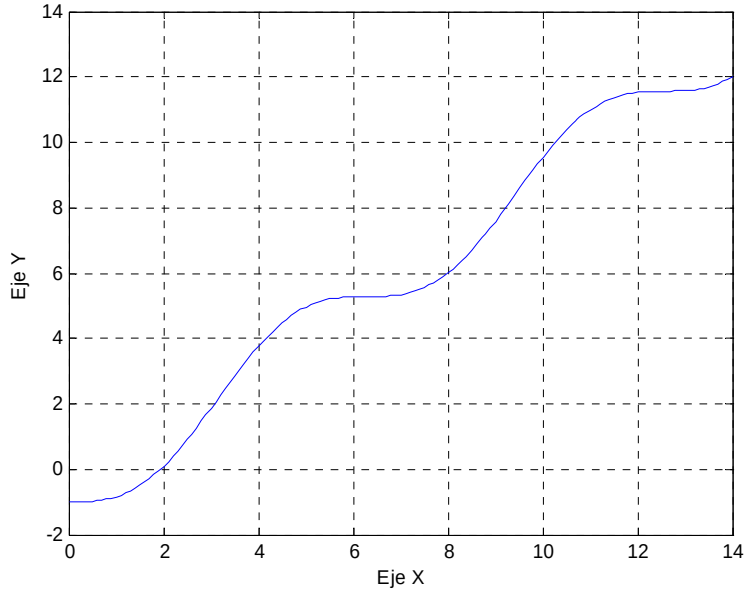
MÓDULO 7

RAÍCES DE UNA FUNCIÓN DE UNA VARIABLE

1 – Escribe una función llamada fun.m que tenga como parámetros de entrada los valores de un vector x, y que tenga como salida y= x-sen(x)-1.

_____

2 – Escribe a continuación un script llamado mifeval.m que defina un vector x, calcule los valores de y en la función anterior (fun.m) y represente la siguiente figura. Ayuda: Utiliza la sentencia y = feval('fun', x). Teclea help feval para más información.



_____
_____
_____
_____

3 – Escribir una función que calcule una raíz de una función por el método de Bolzano Para ello escribe una función (function [x,it]=bisecc(funcion,tol,a,b)) , que admita como parámetros de entrada cualquier función tipo fun.m, la tolerancia tol de la raíz, y el intervalo (a,b) donde se ha de buscar la raíz, y devuelva la raíz x y el número de iteraciones it del método de bisección. Utilizar el comando fprintf('\n %i %f %f %f %f',it, x, y,a,b) para sacar por pantalla los resultados de cada iteración.
Nota: cuando una función entra como parámetro de otra función esta se escribe entre comillas simples, ej. de llamada desde línea de comando :

[x,it]=bisec('fun',0.001,0,4) ó [x,it]=bisec('fun2',0.001,0,4).

La sentencia para ejecutar desde el programa será

fa=feval(funcion,a);