MÉTODOS MATEMÁTICOS PARA LA INGENIERÍA MECÁNICA

PRÁCTICAS

Damián Ginestar Peiró

6 de septiembre de 2006

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO

DEPARTAMENTO DE MATEMÁTICA APLICADA
UNIVERSIDAD POLITÉCNICA DE VALENCIA

Índice general

Práctica 1

Introducción al programa MatLab

1.1. Preguntas más frecuentes

¿ Qué es el MatLab y cuáles son sus aplicaciones?

El MatLab es un sistema interactivo orientado al cálculo matricial. Su nombre es una abreviatura de **Mat**rix **Lab**oratory.

Debido a su versatibilidad permite ser utilizado en multitud de aplicaciones de tipo científico o tecnológico como, por ejemplo, en el campo del procesamiento de la señal o de la imagen, en la simulación de sistemas dinámicos y teoría de control, en el estudio de redes neuronales, etc.

¿ Existe algún comando del MatLab que permita trabajar con comandos o ficheros externos al MatLab?

Sí, para trabajar con funciones definidas por el usuario se puede utilizar el comando `path` que permite ampliar el árbol de directorios donde MatLab busca posibles funciones construidas por el usuario. Para ejecutar comandos o ficheros del DOS se utiliza el signo de admiración **!** seguido del nombre del comando.

¿Distingue MatLab entre mayúsculas y minúsculas?

Hay que tener en cuenta que MatLab distingue entre mayúsculas y minúsculas en los nombres de los comandos, funciones y variables, por lo tanto, hay que tener cuidado en escribirlas correctamente.

¿Se puede guardar la sesión de trabajo en un fichero?

El comando `diary` permite guardar el texto de la sesión. La pauta a seguir es la palabra `diary` seguida de la unidad donde se pretende almacenar la información y el nombre del fichero.txt. Para añadir texto le indicaremos donde empieza el parrafo que queremos guardar con el comando `diary on` y donde termina con el comando `diary off`.

El comando `save` guarda las variables en un fichero denominado **matlab.mat** y estas variables se cargan con el comando `load`.

¿Cómo se utiliza la ayuda?

El comando de ayuda del MatLab se denomina `help`. Se ejecuta seguido del nombre de la función sobre la que se quiere obtener ayuda. Además, si se desea buscar toda la información sobre algún tema utilizaremos el comando `lookfor` seguido de la palabra de la que se pretende obtener la información. En versiones superiores a la 5.2 aparecen otros comandos de ayuda como pueden ser el `helpwin`. Tecleando esta palabra se abre una ventana donde podemos encontrar toda la ayuda clasificada por materias, eligiendo la que se desee y pulsando **ENTER** encontraremos la respuesta buscada.

Con el comando `helpdesk`, se abre una ventana de ayuda con el navegador web que se tenga definido por defecto.

¿Cómo se puede salir del MatLab?

El comando `quit` permite abandonar el MatLab.

¿Cómo se puede detener algún cálculo, gráfico o impresión del MatLab sin salir del paquete?

Con el comando **CTRL-C** o con **CTRL-BREAK.**

¿Qué sucede si una instrucción no cabe en una línea?

Las instrucciones de MatLab se terminan pulsando **ENTER**. Si una instrucción no cabe en una línea, se escriben 3 puntos o más y se puede continuar en la línea siguiente.

¿Cómo se hacen los comentarios?

Los comentarios se hacen con el carácter %. Para que MatLab no muestre el resultado de una instrucción ésta debe terminar con ; .

1.2. Operaciones elementales, variables y constantes

Con el MatLab se puede realizar cualquier operación que podríamos hacer con una calculadora. Cada una de estas operaciones por defecto se guarda en la variable `ans`. Si nosotros no queremos trabajar con esta variable, antes de realizar cualquier operación, deberemos hacer una asignación a la variable donde pretendemos que se guarde el resultado. El proceso consiste en escribir el nombre de la variable seguida del signo = y de la operación que pretendemos realizar. Cada vez que escribamos la variable y apretemos **ENTER**, el programa devolverá su valor. El nombre de la variable se puede utilizar tantas veces como deseemos, pero hay que recordar que guardará sólo el último valor.

Al igual que otros paquetes matemáticos el resultado que obtenemos en pantalla se puede visualizar en diferentes formatos. Si no se indica lo contrario, por defecto se obtienen cuatro cifras decimales. Con el comando `format long` aparecen más cifras decimales. Para volver a su formato estándar escribiremos `format short`. Si se pretende obtener el resultado con un formato racional escribiremos `format rat`.

Si se desea conocer las variables que están activas hasta este momento utilizaremos el comando `who`, si además queremos saber de qué tipo son y su tamaño se utiliza el comando `whos`. Si queremos borrar alguna variable teclearemos el comando `clear` seguido del nombre de la variable que deseamos borrar. Utilizando sólo el comando `clear` se borran todas las variables, que se tienen asignadas en la memoria.

El MatLab tiene una lista de las constantes que frecuentemente aparecen en problemas matemáticos o técnicos, por ejemplo el número **pi**, el número **e** que la escribiremos como **exp**(1) , la unidad imaginaria que la podremos escribir como `i` o `j`, etc.

Así si, por ejemplo, escribimos

```
(2+3*i)+(5-2*i)
```

obtendremos como resultado

```
7+1i
```

Si escribimos

```
(2+3*i)*(5-2*i)
```

o bien

```
(2+3*i)/(5-2*i)
```

obtenemos respectivamente

```
    16+11i
0.1379+0.6552i
```

Como vemos, MatLab simplifica las expresiones con números complejos hasta obtener la forma binómica del número complejo resultado. Por otra parte, el MatLab dispone de algunas funciones específicas para operar con complejos como son las siguientes.

`abs( )`	→	Calcula el módulo del número complejo.
`angle( )`	→	Calcula la fase, en radianes del número complejo.
`real( )`	→	Calcula la parte real del número complejo.
`imag( )`	→	Calcula la parte imaginaria del número complejo.
`conj( )`	→	Calcula el conjugado del número complejo.

Veamos algunos ejemplos

`abs(2+3*i)`	=	3.6056
`angle(2+3*i)`	=	0.9828
`real(2+3*i)`	=	2
`imag(2+3*i)`	=	3
`conj(2+3*i)`	=	2-3i

1.3. Funciones

Al igual que otros paquetes matemáticos, MatLab dispone de un catálogo completo con las funciones más utilizadas. Siempre podremos buscar información de dichas funciones con la ayuda. A modo de ejemplo, a continuación daremos una lista de algunas de estas funciones:

`sqrt( )`	→	raíz cuadrada.
`round( )`	→	redondeo entero más cercano.
`sign( )`	→	función signo.
`exp( )`	→	exponencial.
`log( )`	→	logarítmo neperiano.
`log10( )`	→	logarítmo decimal.
`sin( )`	→	seno.
`cos( )`	→	coseno.
`tan( )`	→	tangente.
`asin( )`	→	arcoseno.
`acos( )`	→	arcocoseno.
`atan( )`	→	arcotangente.

Las funciones hiperbólicas

`sinh( )`	→	seno hiperbólico.
`cosh( )`	→	coseno hiperbólico.
`tanh( )`	→	tangente hiperbólica.
`asinh( )`	→	arcoseno hiperbólico.
`acosh( )`	→	arcocoseno hiperbólico.
`atanh( )`	→	arcotangente hiperbólica.

etc.

1.4. Matrices. Operaciones con matrices

La estructura matricial es la forma natural de trabajar en MatLab. Las matrices 1×1 son escalares, las matrices $1 \times n$ ó $n \times 1$ son vectores filas o columna, respectivamente

Para introducir los elemetos de una matriz se comienza con un corchete. Los elementos de las filas se introducen separándolos entre comas, se indica que finaliza la fila con un punto y coma, finalizaremos el proceso cerrando el corchete.

En el siguiente ejemplo asignamos a la variable `A` una matriz 3×3.

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

Otra forma más visual de introducir los datos de la matriz sin utilizar comas o puntos y coma es separando los elementos de la fila por espacios y escribiendo cada fila en un línea diferente. Así, por ejemplo, podemos escribir

```
B =[1
    2
    3]
```

o

```
C =[1 0 0
    2 0 1
    0 0 2]
```

Una vez que hemos definido estas matrices se puede obtener el elemento i, j de la matriz A escribiendo `A(i,j)`.

Se pueden realizar multiples operaciones con matrices y vectores. Los signos que se utilizan son: para sumar +, para restar -, para multiplicar *, la potenciación se indica con ^ seguido de número al que se pretende elevar la matriz. Todas las operaciones se llevan a cabo entre dos matrices. Si ello no es posible, MatLab devuelve un mensaje de error.

A parte de las operaciones anteriores, MatLab incorpora la división por la izquierda (\ ) y la división por la derecha (/). Así,

$$\begin{aligned} A\backslash B &\rightarrow \text{es equivalente a } A^{-1}B. \\ A/B &\rightarrow \text{es equivalente a } AB^{-1}. \end{aligned}$$

A parte de este tipo de operaciones podemos representar la traspuesta de una matriz con el signo '. El producto escalar y el producto vectorial de dos vectores con los comandos `dot( )` y `cross( )`, respectivamente.

Cuando se quiere hacer operaciones sobre los elementos de las matrices hay anadirles a las matrices un punto. Así, si escribimos

```
A=[1  2  3]
```

y tratamos de calcular `A^2` nos dará un error, ya que el cuadrado de A no está bien definido. Si se pretende obtener una nueva matriz cuyos elementos sean los cuadrados de los elementos de A, podemos escribir


```
A.^2
```

obteniendo como resultado

```
[1 4 9]
```

Este resultado se puede obtener también escribiendo

```
A.*A
```

A parte de acceder a un solo elemento de una matriz, es posible acceder a submatrices de una matriz dada. Por ejemplo, dada

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

si queremos el elemento (1,1) de la matriz escribiremos `A(1,1)` si queremos las dos primeras filas de la tercera columna, escribiremos `A(1:2,3)` y el resultado será

$$\begin{bmatrix} 3 \\ 6 \end{bmatrix},$$

si queremos todos los elementos de la tercera fila escribiremos `A(3,:)` y el resultado obtenido

$$[7\ 8\ 9],$$

si queremos la submatriz formada por la primera y la tercera columna, escribiremos `A(:,[1,3])` el resultado será

$$\begin{bmatrix} 1 & 3 \\ 4 & 6 \\ 7 & 9 \end{bmatrix}.$$

Por último, si escribimos `A([1,3],[1,3])`, el resultado obtenido será

$$\begin{bmatrix} 1 & 3 \\ 7 & 9 \end{bmatrix}.$$

A parte de estas operaciones existen funciones que nos permiten construir matrices, como por ejemplo:

`eye(n)`	→	matriz identidad $n \times n$.
`zeros(n,m)`	→	matriz nula $n \times m$.
`diag(A)`	→	devuelve un vector con la diagonal de `A`.
`diag(x)`	→	devuelve una matriz con `x` en la diagonal.
`triu(A)`	→	devuelve la parte triangular superior de `A`.
`tril(A)`	→	devuelve la parte triangular inferior de `A`.
`rand(n,m)`	→	devuelve una matriz generada aleatoriamente.

A parte se pueden construir matrices por bloques, por ejemplo la intrucción

```
B=[zeros(2,3),eye(2);zeros(3,2),eye(3)]
```

devuelve una matriz de la forma

$$B = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} .$$

1.5. Ejercicios

1. *a*) Si $z_1 = 3 + 5i$ y $z_2 = 1 - 2i$. Calcular

$$|z_1| \,,\ z_1 + z_2,\ z_1 * z_2,\ z_1/z_2 \ .$$

Comprobar que $\arg(z_1/z_2) = \arg(z_1) - \arg(z_2)$.

b) Calcular parte real, imaginaria, módulo y argumento de $\left(1 + \sqrt(3i)\right)^{1-i}$.

c) Calcular

$$(8)^{\frac{1}{5}} \ , \quad e^7 + \ln(5) \,, \quad \frac{\ln(9) + \operatorname{sen}(\pi/5)}{45 - \cos(23)} \ .$$

2. Dadas

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 6 \\ 0 & 1 & 9 \end{pmatrix} \text{ y } b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix},$$

calcular

a) $A + A$,

b) $2A$,

c) AA,

d) A^2,

e) $A\backslash b$ y comprobar que el resultado coincide con $A^{-1}b$,

f) b^T/A y comprobar que el resultado coincide con $b^T A^{-1}$,

g) Calcular el producto escalar $b \cdot b$ y el producto vectorial $b \wedge b$.

3. Introducir las siguientes matrices

$$D = \begin{bmatrix} 1 & -3 & 4 \\ 2 & -5 & 7 \\ 0 & -1 & 1 \end{bmatrix},\ E = \begin{bmatrix} 2 & 4 & 6 \\ 4 & 5 & 6 \\ 3 & 1 & 2 \end{bmatrix},\ c = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}.$$

a) Calcular $D + E, DE, D^T E$.

b) A patir de las matrices D y E anteriores sacar sus vectores fila, sus vectores columna y una submatriz 2×2 formada por las filas 1 y 2 y las columnas 2 y 3.

c) Resolver el sistema

$$E\ x = c$$

d) Construir una matriz diagonal que contenga la diagonal de la matriz E.

e) Construir las matrices por bloques

$$\left[\ D \,\middle|\, E\ \right] \ , \quad \left[\frac{D}{E}\right] \ .$$

4. Dada la matriz

$$E = \begin{bmatrix} 2 & 4 & 6 \\ 4 & 0 & 6 \\ 3 & 1 & 2 \end{bmatrix}$$

obtener su parte estríctamente triangular inferior (sin la diagonal) y su parte estríctamente triangular superior.

Práctica 2

Bucles y funciones

2.1. Bucles

La utilización de bucles permite realizar múltiples operaciones que se repiten, mediante instrucciones muy sencillas. El bucle más sencillo que se puede escribir es un bucle tipo `for` y tiene la forma

```
for i =1:n
      instrucciones
end
```

Por ejemplo, si se desea calcular la suma de las diez primeras potencias del número 2, es decir, si se pretende calcular

$$\sum_{i=1}^{10} 2^i,$$

se pueden utilizar las siguientes instrucciones:

```
suma=0;
for i=1:10
    suma=suma+2^i;
end
suma
```


Si lo que se pretende es calcular

$$\prod_{j=1}^{10} \frac{j}{j+1},$$

las instrucciones a utilizar serían

```
produc=1;
for j=1:10
   produc=produc*j/(j+1);
end
produc
```

En algunas ocasiones se necesita que el contador no varíe de uno en uno. Para ello, sólo tendremos que indicar el tamaño de paso en la instrucción **for.** Por ejemplo, las siguientes instrucciones

```
x=[ ];
for i=10:-1:1
      x=[x,i^2];
end
x
```

construyen un vector cuyas componentes son los cuadrados de los primeros diez números naturales en orden inverso.

Hay que tener en cuenta que MatLab dispone de operaciones matriciales y vectoriales optimizadas y hay que evitar el uso de bucles en lo posible, ya que hacen que los programas funcionen más lentamente.

Otro tipo de bucles utilizan el comando `while`. La estructura de estos bucles es la siguiente

```
while relacion
    instrucciones
end
```

Por ejemplo, si se quiere calcular el mayor número entero n tal que $2^n < 3000$, se puede hacer utilizando las siguientes instrucciones

```
n=0
while  2^n < 3000
    n=n+1;
end
n-1
```

Otra estructura útil es la estructura `if` que se utiliza del siguiente modo:

```
if  relacion
   instrucciones
end
```

Además se pueden hacer ramificaciones como se muestra en el siguiente ejemplo.

```
if  n<0
    paridad=0;
elseif rem(n,2)==0
  paridad=2;
else
  paridad=1;
end
```

Los operadores que permiten establecer relaciones son:

$<$, $>$, $<=$, $>=$, $==$,
$\sim =$ significa no igual,
& significa y
$\sim$ significa no
| significa o

Por ejemplo, la negación $A \sim = B$ donde A y B son matrices, sólo se cumplirá cuando todos los elementos de A sean distintos de los de B.

Otra estructura de interés es la estructura `switch`, que se utiliza como se muestra en el siguiente ejemplo:

```
switch method
case {1,2}
   disp('metodo lineal')
```



```
case 3:
   disp('metodo cubico}
otherwise:
   disp('metodo de orden superior')
end
```

2.2. Ficheros.m

Matlab dispone, en general de un editor que permite crear ficheros de texto. Estos ficheros suelen llevar la extensión **.m** y permite crear programas, que no son más que un conjunto de instrucciones y funciones. Para ejecutar estos programas bastará incluir el directorio donde se tiene el fichero en el **path** del MatLab, haciendo uso de la función `path()` y teclear el nombre del fichero sin extensión.

Las funciones tienen una estructura especial, como se muestra en el siguiente ejemplo:

```
function s=suma5(x)
% suma5 es una funcion que  suma 5 a  x
      s=x+5;
```

Esta estructura empieza con la palabra `function`. La frase comentada que hay en la línea siguiente a la cabecera es la que aparece al utilizarse el comando de ayuda, es decir, `help suma5`. Una vez definida la función, deberá guardarse en un fichero de nombre igual al nombre de la función con la extensión **.m**, para nuestro ejemplo el nombre del fichero sería `suma5.m`.

Otro ejemplo de función sencilla es el siguiente:

```
function [media,desv]=estad(x)
[m,n]=size(x);
if  m==1
m=n;
end
media =sum(x)/m;
desv =sqrt(sum(x.^2)/m-media^2);
```

Para escribir la desviación típica se ha tenido en cuenta que

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \overline{x}^2.$$

Además el paquete MatLab tiene muchas funciones propias, tanto en el núcleo del programa como en una serie de librerias especializadas llamadas toolboxes, como ejemplo, mostramos algunos ejemplos de funciones que permiten escribir mensajes al usuario.

La instrucción `disp('mensaje')` muestra un mensaje al ususario.

La instrucción `error('mensaje')` devuelve un mensaje y para la ejecución de la función.

La instruccción

```
iter=input('introducir el numero de iteraciones')
```

muestra un mensaje y espera a que se introduzca un valor mediante el teclado que se asigna a la variable `iter`.

2.3. Ejercicios

1. Calcular, utilizando la instrucción `sum( )`

$$\sum_{i=1}^{10} 2^i \ .$$

Y, utilizando la instrucción `prod( )`

$$\prod_{j=1}^{10} \frac{j}{j+1} \ .$$

2. *a*) Escribir un bucle para calcular la suma de los cuadrados de los 25 primeros números naturales.

   b) Calcular el mayor número natural que satisface que

$$3^n + \ln(n) < 2000.$$

   c) Calcular el producto de los 10 primeros números impares

3. Construir una función que calcule la nota media del expediente académico de un alumno de primer curso, a partir de un vector donde estén almacenadas las notas.

4. Si suponemos una versión simplificada del sistema de suspensión de un automovil, el comportamiento entrada-salida, considerando el movimiento del cuerpo sólo en la dirección vertical, viene representado por la función de transferencia

$$F(s) = \frac{\frac{b}{m}s + \frac{k}{m}}{s^2 + \frac{b}{m}s + \frac{k}{m}} \ ,$$

donde b es el amortiguamiento, k la elasticidad del muelle y m la masa. Utilizando MatLab, definir esta función de transferencia. Calcular los valores de la función de transferencia si tomamos $b = 1$, $m = 2$, $k = 3$ y se hace variar la variable s de 0 a 100 de una unidad en una unidad, guardando el resultado en un vector.

5. Escribe un pequeño programa que pida un número por el teclado y compruebe su tamaño. Si el número es menor que 100, el programa debe escribir "numero pequeño", y si el número es mayor o igual que 100, el programa debe escribir algo apropiado. Introduce el código en un bucle para que se puedan introducir varios números, uno detrás de otro.

6. La siguiente expresión

$$\log(n!) \approx n\log(n) - n \ ,$$

constituye la aproximación de Stirling de $\log(n!)$. Escribe un programa para comprobar la validez de esta aproximación para $n = 100$, $n = 1000$ y $n = 5000$, teniendo en cuenta que 5000! es un número demasiado grande para poder ser tratado de forma standard por el ordenador.

7. La ecuación de Van der Waals para los gases es una generalización de la ecuación

$$PV = nRT \ ,$$

que tiene en cuenta la desviación del comportamiento ideal de los gases. Esta ecuación es

$$\left(P + \frac{n^2 a}{V^2}\right)(V - nb) = nRT \ ,$$

donde n es el número de moles del gas, P es la presión en (Pa), R es la constante de los gases (8.314 J/ (mol K)), y a y b son constantes que miden la desviación de la idealidad en el comportamiento del gas. Reordena la ecuación de forma que P se exprese como una función de V, y escribe un pequeño programa que genere una tabla con la presión correspondiente a 10 volúmenes igualmente espaciados entre V_1 y V_2. Las variables a, b, T, V_1 y V_2 han de introducirse por el teclado. Suponed $n = 1$.

8. Dada una matriz A $n \times m$, escribe una función que permute las filas k e i de la matriz A cuya llamada sea `B = permuta(A,k,i)`.

9. Calcula aproximadamente el valor de π utilizando el siguienete resultado

$$\frac{\pi^2 - 8}{16} = \sum_{n=1}^{\infty} \frac{1}{(2n-1)^2(2n+1)^2} \ .$$

¿ Cúantos términos de la serie hace falta sumar para obtener una precisión de $1\,10^{-12}$?.

10. Los números de Fibonacci se calculan haciendo uso de la relación

$$F_n = F_{n-1} + F_{n-2} \ ,$$

con $F_0 = F_1 = 1$

a) Calcula los 10 primeros números de Fibonacci.

b) Para los 50 primeros números de Fibonacci calcula el cociente F_n/F_{n-1}. Este cociente se aproxima a valor del *número aureo*$(1 + \sqrt{(5)})/2)$. ¿ Qué puedes decir de tus resltados?

Práctica 3

Gráficas con Matlab

El paquete MatLab permite obtener las gráficas de cualquier función matemática tanto si representa una curva plana o una superficie. Además, permite agrupar y superponer gráficas. Otras opciones típicas de los programas de gráficos como colores, marcos, etc. se pueden utilizar en este paquete.

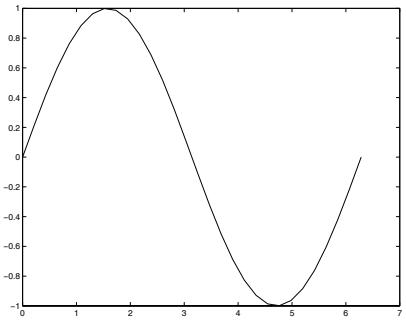
3.1. Gráficas 2D

La representación gráfica 2D de una función se puede obtener cuando la función se expresa en coordenadas cartesianas o paramétricas.

El comando `plot(x,y)` representa los pares que tienen como abcisas los elementos del vector `x` y como ordenadas los elementos del vector `y`. Con el comando `plot(y)` toma como abcisas los números naturales 1,2, ..., n. El comando `linspace(a,b,N)` genera `N` puntos igualmente espaciados comprendidos entre `a` y `b`. Así, para la generación de gráficas de funciones se procede, por ejemplo, del siguiente modo:

```
x=linspace(0,2*pi,30);
y=sin(x);
plot(x,y)
```

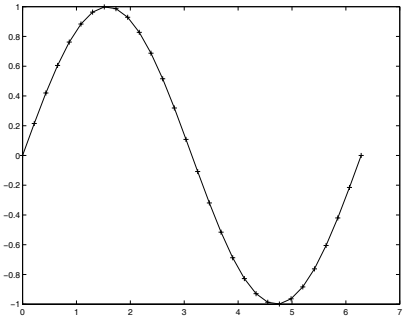
obtenemos la siguiente gráfica



Es posible generar la misma gráfica con líneas y cruces escribiendo

```
plot(x,y,x,y,'+')
```

con lo que se obtiene:



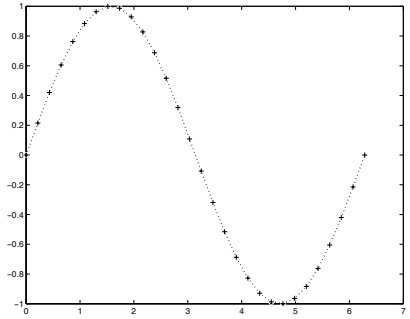
Por otra parte, se pueden controlar los colores y los estilos de las líneas utilizando los símbolos de la siguiente tabla:

Símbolo	Color	Símbolo	Estilo
y	amarillo	.	puntos
m	magenta	o	círculos
c	cyan	x	aspas
r	rojo	+	cruces
g	verde	*	estrella
b	azul	-	línea
w	blanco	:	línea de puntos
k	negro	- .	línea y puntos
		- -	línea de guiones

Por ejemplo, se puede escribir:

```
y=sin(x)
plot(x,y,'g:',x,y,'wo',x,y,'r:',x,y,'c+')
```

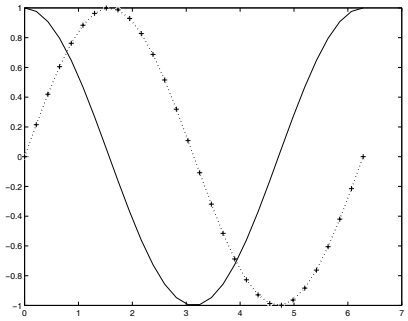
obteniéndose



Cada vez que se ejecuta el comando `plot` desaparece la figura anterior. Si pretendemos superponer gráficas el comando `hold` nos permite mantener la gráfica de la función donde estamos trabajando. Por ejemplo, si escribimos:

```
plot(x,y,'g:',x,y,'wo',x,y,'r:',x,y,'c+')
hold on
z=cos(x)
plot(x,z)
```

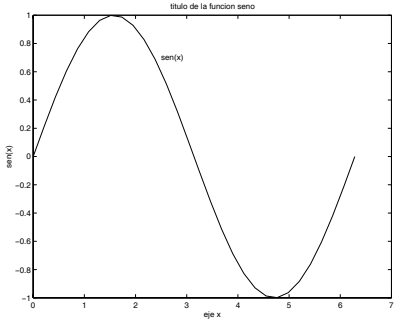
se obtiene la gráfica siguiente:



Hay distintas funciones que controlan la apariencia del gráfico. Así la función `grid on` introduce un mallado del gráfico y `grid off` quita el mallado. Las fuciones `xlabel` e `ylabel` generan un título para el eje x y el eje y, respectivamente. La función `title` genera un título para el gráfico. La función `text` permite poner texto en una zona del gráfico. Un ejemplo para la utilización de estas funciones es el siguiente

```
plot(x,y)
title('titulo de la funcion seno')
xlabel('eje x')
ylabel('sen(x)')
text(2.5,0.7,'sen(x)')
```

obteniendo la siguiente gráfica:



La función `axis` tiene control sobre la apariencia de los ejes, así, por ejemplo:

```
axis([xmin, xmax, ymin, ymax])
```

fija el eje x y el eje y de forma que el valor mínimo para las x-s sea `xmin`, el valor máximo para las x-s sea `xmax`, el valor mínimo para las y-s sea `ymin`, el valor máximo para las y-s sea `ymax`.

`axis auto`: devuelve la escala a los valores de defecto.

`axis equal`: usa la misma escala para las x-s que para las y-s.

`axis normal`: devuelve la escala a sus valores de defecto.

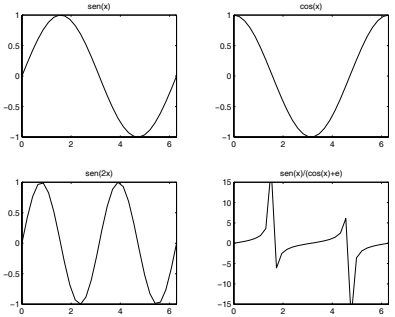
`axis off`: no dibuja los ejes.

`axis on`: vuelve a dibujar los ejes.

El comando `subplot` permite dividir la zona de dibujo en zonas y en cada zona dibujar una curva distinta. Su funcionamiento se muestra en el siguiente ejemplo

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
a=2*sin(x).*cos(x);
b=sin(x)./(cos(x)+eps);
subplot(2,2,1) % se divide la zona de dibujo
       % en 2 x 2 graficos y se selciona
       % la primera zona (arriba izquierda).
plot(x,y)
axis([0, 2*pi, -1, 1])
title('sen(x)')
subplot(2,2,2) % se seleciona la segunda zona
plot(x,z)
axis([0, 2*pi, -1, 1])
title('cos(x)')
subplot(2,2,3) % se seleciona la tercera zona
plot(x,a)
axis([0, 2*pi, -1, 1])
title('sen(2x)')
```

```
subplot(2,2,4) % se seleciona la cuarta zona
plot(x,b)
axis([0, 2*pi, -15, 15])
title('sen(x)/(cos(x)+e'))
```



El Matlab permite también realizar gráficas usando escalas logarítmicas y semilogarítmicas. Así,

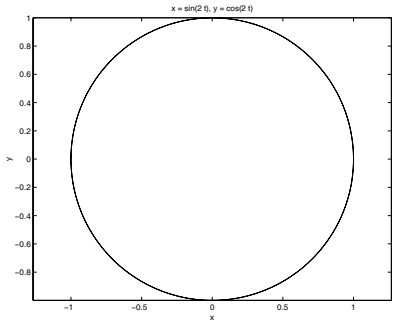
`loglog` : Es una función igual que `plot` pero usa escalas logarítmicas para el eje x y el eje y.

`semilogx` : Es una función igual que `plot` pero usa escala logarítmica para el eje x y lineal para el eje y.

`semilogy` : Es una función igual que `plot` pero usa escala logarítmica para el eje y y lineal para el eje x.

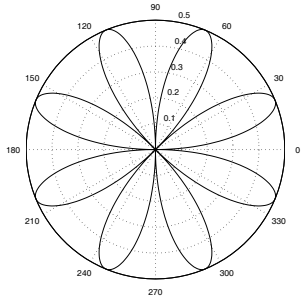
Si pretendemos dibujar curvas que vienen representadas en coordenadas paramétricas podremos utilizar el comando `ezplot` la instrucción sería como sigue:

```
ezplot('sin(2*t)','cos(2*t)',[0,2*pi])
```

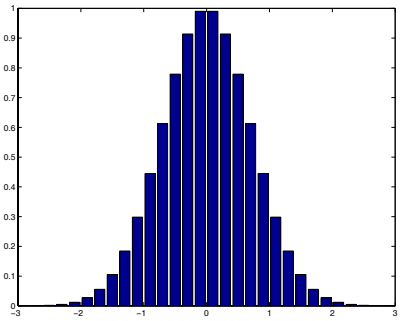
Una gráfica de una función en coordenadas polares puede crearse usando la función `polar`, como muestra el siguiente ejemplo:

```
t=0:0.01:2*pi;
r=sin(2*t).*cos(2*t);
polar(t,r)
```



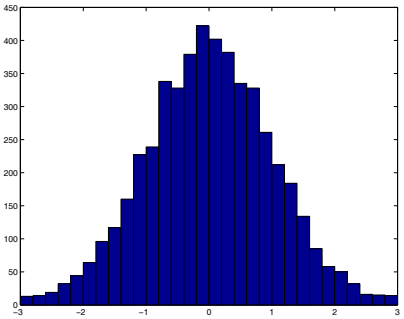
La función `bar` genera un diagrama de barras. Un ejemplo del diagrama de barras asociado a una gaussiana es el siguiente:

```
x=-2.9:0.2:2.9; % especifica el numero de divisiones
y= exp(-x.*x);
bar(x,y)
```



La función `hist` genera el histiograma asociado a los datos contenidos en un vector. Un ejemplo de su uso para un vector de números aleatorios distribuidos normalmente es

```
x=-2.9:0.2:2.9; % especifica el numero de divisiones
y= randn(5000,1);
hist(y,x)
```



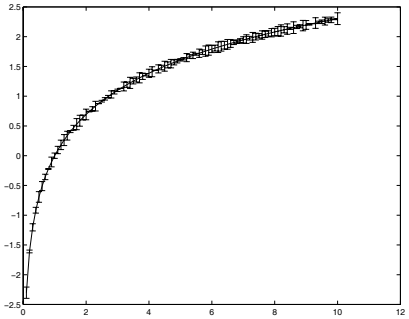
¿Qué pasa si se sustituye la función `randn` por la función `rand`?.

Se pueden dibujar gráficas con barras de error. Para ello, se usa la función `errorbar`. Un ejemplo de su uso es el siguiente

```
x=0.1:0.1:10;
y= log(x);
```



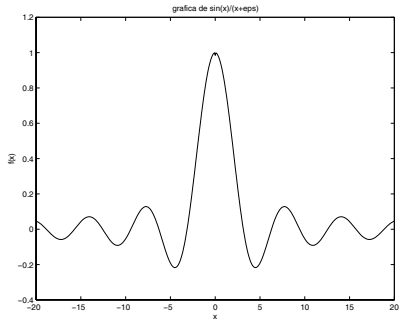
```
e=rand(size(x))/10;  % crea un vector de errores aleatorio
errorbar(x,y,e)
```



La función `fplot` permite generar la gráfica de una función de una variable, sin necesidad de generar vectores con los datos. La sintaxis es de la forma `fplot('fun',[xmin xmax])` o bien `fplot('fun',[xmin xmax ymin ymax])`.

Un ejemplo de su uso es el siguiente:

```
fplot('sin(x)./(x+eps)',[-20 20 -0.4 1.2]);
title('grafica de sin(x)/(x + eps)');
xlabel('x')
ylabel('f(x)')
```



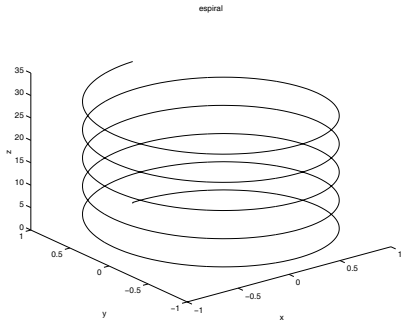
3.2. Gráficos 3D

Para generar la representación de una curva en el espacio, se puede utilizar la función `plot3`. Supongamos que se quiere dibujar la espiral

$$E = \begin{cases} x(t) = \cos(t) \\ y(t) = \text{sen}(t) \\ z(t) = t \ , \quad t \in [0, 10\pi] \ . \end{cases}$$

Para ello, se puede escribir

```
t=0:pi/50:10*pi;
x=sin(t);
y=cos(t);
z=t;
plot3(x,y,z,'r');
title('espiral')
xlabel('x'), ylabel('y'),zlabel('z')
```



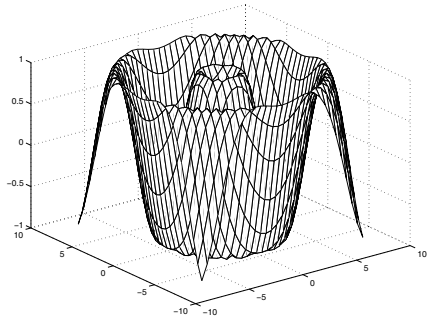
Se puede controlar la escala de los ejes mediante la función `axis([xmin, xmax, ymin,ymax,zmin,zmax])` de forma similar a como se hace para los gráficos bidimensionales. También la función `text(x,y,z,'texto')` nos permite escribir un texto en las coordenadas (x, y, z) del gráfico.

Supongamos ahora que se quiere representar una superficie $Z = z(x, y)$. Para ello, existen distintas funciones en Matlab que permiten hacer distintas representaciones. Consideremos la función

$$z(x, y) = \text{sen}\left(\sqrt{x^2 + y^2}\right) \ .$$

Una representación de esta función puede hacerse como se muestra en el siguiente ejemplo:

```
x=-7.5:0.5:7.5;
y=x;
[X,Y]=meshgrid(x,y);
Z=sin(sqrt(X.^2+ Y.^2));
mesh(X,Y,Z)
```



Otra posibilidad es escribir

```
mesh(Z)
```

Una alternativa a la función `mesh` es la función `surf`. Así, se puede usar

```
surf(X,Y,Z)
```

Se pueden generar curvas de nivel de una determinada función mediante las funciones `contour` y `contour3`.

```
contour(X,Y,Z,20)
```

genera 20 curvas de nivel de la función $z(x, y)$ en el plano y

```
contour3(X,Y,Z,20)
```

da la representación de estas curvas en el espacio.

Se puede obtener una representación bidimensional mediante colores de una función $z(x, y)$ mediante la función `pcolor`. Así,

```
pcolor(Z)
```

produce esta representación con el juego de colores que tiene Matlab por defecto. El juego de colores que usa Matlab se puede cambiar mediante la función `colormap`. Distintas posibilidades de colores que tiene matlab implementadas se muestran en la siguiente tabla

función	Descripción
hsv	Saturación de tonos
hot	Negro, rojo, amarillo y blanco
pink	Sombras pastel rosa
gray	Escala de grises
bone	Escala de grises con matices azules
jet	Una variante de hsv
copper	Tonos cobre
prism	prisma
flag	rojo, blanco, azul y negro alternativos

Se pueden combinar estas funciones de la siguiente manera:

```
colormap(hot)
pcolor(X,Y,Z)
shading flat   % quita la rejilla
hold on
contour(X,Y,Z,20,'k')
hold off
```

Se puede cambiar el punto de vista de las representaciones tridimensionales de funciones mediante el comando `view`. Una posibilidad es pasarle a la función el azimut y la elevación en grados de la direccion en la que se quiere mirar

```
view(phi,theta)
```

o bien se la pasa un vector en esa dirección

```
view([x1,x2,x3])
```


3.3. Ejercicios

1. Representar gráficamente las siguientes curvas:

   a) $y = x^5 + 3x + 5, \quad x \in [0, 1]$,

   b) $y = e^{3x} + \operatorname{sen}(x)$, $x \in [-0{,}5, 0{,}5]$,

   c) $\begin{cases} x(t) = t - \operatorname{sen}(t) \\ y(t) = t - \cos(t) \end{cases}$, $t \in [0, 4\pi]$.

   d) $\rho = \cos(\theta) + \operatorname{sen}(\theta/4)$, $\theta \in [0, 2\pi]$.

2. Los polinomios de Legendre se definen mediante la siguiente relación de recurrencia

$$(n+1)P_{n+1}(x) - (2n+1)xP_n(x) + nP_{n-1}(x) = 0 \ ,$$

   con $P_0(x) = 1$, $P_1(x) = x$ y $P_2(x) = (3x^2 - 1)/2$. Calcula los tres polinomios de Legendre siguientes y dibuja los 6 primeros polinomios de Legendre en el intervalo $[-1, 1]$.

3. Utilizando el comando adecuado divide la ventana de gráficos en cuatro ventanas y dibuja en cada una de estas vetanas, una de las siguientes funciones:

$$\begin{aligned} y &= \tan(x) \ , \quad x \in [0, 0{,}5] \ , \\ y &= \cosh(5 * x) \ , \quad x \in [0, 10] \ , \\ y &= e^{2x} + 5\operatorname{sen}(2x) \ , \quad x \in [0, \pi] \ , \\ y &= x^2 + 3x + 4, \ x \in [0, 1] \ . \end{aligned}$$

4. Averigua, aproximadamente, el punto de corte de las siguientes funciones en el intervalo [0,2]

$$\begin{aligned} y &= 2x^3 + 5{,}4x^2 + 4{,}8x + 1{,}4 \ , \\ y &= 7x + 10 \ . \end{aligned}$$

5. Obtener una representación de la superficie

$$z(x, y) = \sqrt{x^2 + y^2} \ ,$$

   para $(x, y) \in [-1, 1] \times [-1, 1]$.

6. Dibuja la gráfica de la superficie

$$z(x, y) = x^2 + y^2 \ ,$$

   en el dominio $[-1, 1] \times [-1, 1]$ y superpón a la representación 5 curvas de nivel.

Práctica 4

Raíces de ecuaciones, integrales y ecuaciones diferenciales

En esta práctica presentaremos algunas funciones disponibles en Matlab para realizar cálculos aproximados.

4.1. Búsqueda de raíces

Supongamos que se buscan raíces de la función

$$f(x) = \frac{1}{(x-0{,}3)^2+0{,}01} + \frac{1}{(x-0{,}9)^2+0{,}4} - 6 \ .$$

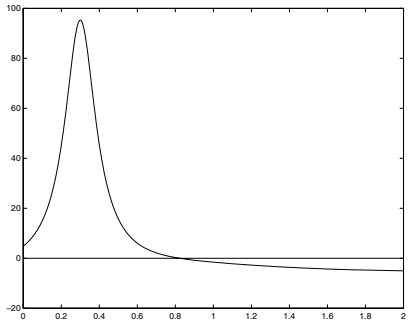
Primero se dibuja la función junto con el eje de las x-s, para ello, primero se define la función

```
function y=func(x)
%  funcion de la que se quiere buscar la raiz
y =1/((x-0.3).^2 +0.01) +  1/((x-0.9).^2 + 0.4)- 6;
```

y se utiliza la función `fplot( )`, del siguiente modo:

```
fplot('func', [0 2])
hold on
fplot('0', [0 2], 'g')
hold off
```


obteniendo



Observamos que tiene un cero cerca de $x = 0{,}8$. Se puede mejorar la precisión de esta raíz con la función `fzero( )`, así escribimos

```
fzero('func', 0.8)
```

obteniendo el valor $x = 0{,}8222$.

Supongamos ahora que queremos obtener una raíz del sistema de ecuaciones

$$\begin{aligned} x_1^2 - 10x_1 + x_2^2 + 8 &= 0 \ , \\ x_1 x_2^2 + x_1 - 10x_2 + 8 &= 0 \ . \end{aligned} \tag{4.1}$$

La siguiente función de Matlab es una implementación del método de Newton para sistemas de ecuaciones,

```
function [xr,k]=newtonsi(x,tol,imax)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Metodo de Newton para sistemas de ecuaciones
%  Uso: [xr,k]=newtonsi(x,tol,imax)
%
%  Input:
% x = vector x1,x2,...,xn inicial,
% tol=tolerancia
%
%  Se ha de disponer de las funciones:
```

```
%      f.m  funcion y=f(x) donde se define el sistema
%      jac.m funcion df=jac(x) donde se define la matriz
%     derivada del sistema.
%
%  Output: xr= raiz,  k= numero de iteraciones.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k=1;
epi=1;
x1=x;
while norm(epi)>tol
   x=x1;
   fxn=f(x);
   axn=jac(x);
   epi=axn\fxn';
   x1=x-epi';
   k=k+1;
   if k>imax
      disp('no converge')
      break
   end
end
xr=x1;
```

Esta función usa dos funciones auxiliares, `f.m` y `jac.m` donde se definen la función que define el sistema de ecuaciones y la matriz derivada de esta función, respectivamente. Para el problema (4.1) estas funciones son de la forma

```
function y=f(x)
% funcion para utilizar con newtonsi.m
y(1)=x(1)^2-10*x(1)+x(2)^2+8;
y(2)=x(1)*x(2)^2+x(1)-10*x(2)+8;
```

y

```
function df=jac(x)
% matriz jacabiana para usar con newtonsi.m
df(1,1)=2*x(1)-10;
df(1,2)=2*x(2);
df(2,1)=x(2)^2+1;
df(2,2)=2*x(1)*x(2)-10;
```


Una vez se escriben estas funciones en sus correspondientes ficheros, podemos ejecutar la instrucción `[xr,k]=newtonsi([0 0],1.e-5,100)`, obteniendo como resultado la raíz `xr`= [1,00, 1,00] en `k`= 10 iteraciones.

4.2. Integración aproximada

Para el cálculo aproximado de integrales definidas el Matlab tiene implementadas las funciones `trapz`, `quad` y `quad8`.

Si queremos calcular

$$\int_0^{\frac{\pi}{2}} \text{sen}(x)\, dx \ ,$$

podemos escribir

```
x=0:pi/10:pi/2;
y=sin(x);
integral=trapz(x,y)
```

La función `trapz( )` utiliza el método de los trapecios basado en la división utilizada para definir el vector `x`.

Otras posibilidades son

```
integral=quad('sin',0,pi/2)
```

o bien,

```
integral=quad8('sin',0,pi/2)
```

4.3. Ecuaciones diferenciales

El Matlab tiene implementadas, entre otras, las funciones `ode23` u `ode45`, que permiten resolver problemas de valores iniciales asociados a ecuaciones diferenciales ordinarias.

Supongamos que se quiere resolver un problema de valores iniciales asociado a un oscilador de Van der Pol

$$\frac{d^2x}{dt^2} - \nu(1-x^2)\frac{dx}{dt} + x = 0 \ .$$

Primero se ha de expresar la ecuación diferencial en forma normal, esto es, se introducen las variables $y_1 = x$ y $y_2 = \frac{dx}{dt}$, con lo que se tiene el sistema de ecuaciones

$$\begin{aligned} \frac{dy_1}{dt} &= y_2 \ , \\ \frac{dy_2}{dt} &= \nu(1-y_1^2)y_2 - y_1 \ . \end{aligned}$$

Luego en un fichero se contruye una función con la siguiente estructura

```
function yprime=vdpol(t,y)
% devuelve las derivadas del oscilador de Van der  Pol
%  se escoge mu = 2
mu=2;% se suele elegir 0 < mu < 10
yprime = [ y(2);
  mu*(1-y(1)^2)*y(2) - y(1) ] ;
```

Luego se puede calcular el problema de valores iniciales asociado al oscilador de Van der Pol con las condiciones iniciales $y_1(0) = 1$, $y_2(0) = 0$ para $t \in [0, 30]$ de la forma siguiente

```
[t,y]=ode23('vdpol',[0,30],[1;0]);
```

y para dibujar las soluciones hacemos lo siguiente:

```
y1=y(:,1);
y2=y(:,2);
plot(y1,y2) % diagrama de fases
```

El funcionamiento de la función `ode45` es totalmente similar.

4.4. Ejercicios

1. Los problemas relacionados con la cantidad de dinero requerida para pagar una hipoteca en un periodo fijo de tiempo involucran una fórmula de la forma,

$$A = \frac{P}{i}\left[1 - (1+i)^{-n}\right] \ ,$$

conocida como *ecuación de anualidad ordinaria*. En esta ecuación, A es la cantidad de la hipoteca, P es la cantidad de cada pago, e i es la tasa de interés por periodo para los n periodos de pago.

Suponed que se necesita una hipoteca de 120.000 euros de una casa sobre un periodo de 30 años, y que la persona que pide el préstamo no puede hacer pagos de más de 1500 euros al mes. ¿Cuál es la tasa máxima de interés que esta persona puede pagar?.

2. Obtén las raíces de la ecuación

$$x = x^2 \operatorname{sen}(x) \ ,$$

comprendidas en el intervalo $[0, 10]$.

3. Encontrar la raíz positiva más pequeña de la ecuación

$$\cos(3x) = \cos(7x) \ ,$$

en el intervalo [0,5, 1,5] con una precisión de seis cifras decimales.

4. la torsión, T, y el esfuerzo cortante máximo, $\tau_{\max}$, para un tubo de radio interno R_i y radio externo R_e, están relacionados por la ecuación

$$TR_e = \frac{\pi}{2}\tau_{\max}\left(R_e^4 - R_i^4\right) \ .$$

Si $R_i = 0{,}2$, encontrar R_e para $\tau_{\max} = 36$ y $T = 0{,}9$.

5. Un objeto de masa M se deja caer verticalmente desde una altura S. Se sabe que al cabo de t segundos la altura del objeto es

$$S(t) = S_0 + \frac{Mg}{k}t - \frac{M^2 g}{k^2}\left(1 - e^{-\frac{kt}{M}}\right) \ ,$$

donde $g = -9{,}8$ m/s^2 y k representa el coeficiente de rozamiento con el aire. Si $M = 0{,}1$ kg, $S_0 = 9{,}2$ m y $k = 0{,}15$ kg s/m. Calcular con una precisión de 10^{-2}s en qué momento el objeto estará a 2.3 m de tierra.

6. El volumen V de un gas en función de la presión P ejercida por un pistón viene dada en la tabla siguiente

P	60	80	100	120	140	160	180
V	80.0	69.2	60.0	52.0	45.0	38.6	32.5

Calcula el trabajo necesario para disminuir V de 80 a 32.5,

$$W = \int_{32,5}^{80} P\, dV \ .$$

7. Se tiene que construir una hoja de techo corrugado usando una máquina que comprime una hoja plana de aluminio convirtiéndola en una cuya sección transversal tiene la forma de una onda senoidal. Supongamos que se necesita una hoja corrugada de 50 cm de longitud y que cada ondulación tiene una altura de un cm respecto de la línea horizontal y un periodo de 2π cm. El problema de encontrar la longitud de la hoja inicial viene dado por la integral

$$L = \int_0^{50} \sqrt{1 + \cos^2(x)}\, dx \ .$$

Estimad esta longitud.

8. Calcular aproximadamente el valor de

$$\int_0^{\infty} \frac{\cos(2x)}{\cosh(x)} \ ,$$

y compararlo con el valor exacto, $\frac{\pi}{2}\,\mathrm{sech}(\pi)$.

9. Sea $z(t)$ (t en minutos) la temperatura de un objeto que está en una habitación que se mantiene a una temperatura de 25° C. Según la ley de enfriamiento de Newton, z verifica la siguiente ecuación diferencial

$$z'(t) = -0{,}02(z(t) - 25) \ .$$

Si al principio el objeto tenía una temperatura de 55° C, se pide:

a) ¿ Qué temperatura tendrá el objeto transcurridos 50 minutos?.

b) ¿ Cúando alcanza el objeto la temperatura de 46° C ?.

10. La intensidad de un circuito RCL con una fuerza electromotriz $E(t)$ viene dada por la ecuación diferencial

$$L\frac{dI}{dt} + RI + \frac{1}{C}Q = E(t) \ .$$

Supongamos que se tiene una resistencia de 2Ω, un condensador de 1 F y una autoinducción de 0.5 H. Si la fuerza electromotriz del circuito es de la forma $E(t) = 5\,\mathrm{sen}(t)$. Sabiendo que se cumple la relación

$$\frac{dQ}{dt} = I \ ,$$

que el condensador en el tiempo $t = 0$ está descargado, y que inicialmente no pasa corriente por el circuito, obtener la evolución de la intensidad en una rama del circuito en función del tiempo desde $t = 0$ a $t = 1$ segundos, y la carga del condensador a los 0.5 segundos.

11. La ecuación de un péndulo físico es de la forma

$$\frac{d^2\theta}{dt^2} + \mathrm{sen}\,(\theta) = 0$$

y una aproximación de esta ecuación es la ecuación del péndulo simple

$$\frac{d^2\theta}{dt^2} + \theta = 0 \ .$$

Compara la evolución temporal del ángulo θ, desde $t = 0$ hasta $t = 10$, con las condiciones iniciales: a) $\theta(0) = \frac{\pi}{6}$, $\frac{d\theta}{dt}(0) = 0$. b) $\theta(0) = \frac{\pi}{3}$, $\frac{d\theta}{dt}(0) = 0$.

Práctica 5

Resolución de sistemas de ecuaciones lineales

Como se ha visto en la teoría, hay dos tipos de métodos para obtener una solución numérica de un sistema de ecuaciones, los métodos directos y los métodos iterativos. Los métodos directos son, generalmente, variantes del método de Gauss y para utilizarlos es necesario conocer explícitamente los elementos de la matriz. Los métodos iterativos obtienen una aproximación numérica tras un número finito de iteraciones. Para utilizar estos métodos se suficiente conocer un método para realizar el producto de la matriz de coeficientes por un vector.

5.1. Métodos directos

Los métodos directos para la resolución de ecuaciones están implementados en el núcleo de Matlab, de forma que resulten lo más eficientes posibles para clases generales de matrices.

La manera más usual de resolver un sistema de ecuaciones en Matlab es haciendo uso del operador . Así, por ejemplo para resolver el sistema

$$\begin{aligned} x_1 + x_2 + 3x_4 &= 0 \\ 2x_1 + x_2 - x_3 + x_4 &= 1 \\ 3x_1 - x_2 - x_3 + 2x_4 &= -3 \\ -x_1 + 2x_2 + 3x_3 - x_4 &= 4 \end{aligned}$$

se introducen las instrucciones


```
A = [ 1, 1, 0, 3 ;
      2, 2, -1, 1;
      3, -1, -1, 2;
      -1, 2, 3,-1];
b=[0;1;-3;4];
x=A\b
```

obteniendo como resultado $x = (-1, 2, 0, 1)^T$.

El operador tiene implementados distintos algoritmos de forma que

- Si A es una matriz triangular usa una sustitución regresiva o progresiva para resolver el sistema.
- Si A es simétrica trata de calcular una descomposición de Cholesky de A.
- Si A no es simétrica o la descomposición de Choleski falla, se usa un algoritmo de Gauss con pivotación parcial.

De una forma similar, podemos usar el operador para resolver simultáneamente varios sistemas de ecuaciones con la misma matriz de coeficientes. Así, por ejemplo, podemos escribir

```
A=pascal(3);
B=magic(3);
x=A\B
```

obtenemos,

```
x=
     19   -3   -1
    -17    4   13
      6    0   -6
```

que es el resultado $x = A^{-1}B$, que se obtiene sin necesidad de calcular A^{-1}.

5.2. Descomposición LU y factorización de Cholesky

El Matlab permite obtener la descomposición LU de una matriz A mediante la función `lu( )`. Así podemos escribir

```
A=magic(3);
[L,U]=lu(A)
```

obteniendo como resultados

```
L=
    1.0000         0         0
    0.3750    0.5441    1.0000
    0.5000    1.0000         0

U=
    8.0000    1.0000    6.0000
         0    8.5000   -1.0000
         0         0    5.2941
```

Por otra parte, ya hemos visto que una matriz simétrica y definida positiva admite una factorización de Cholesky, o sea, una factorización

$$A = LL^T = R^T R \ .$$

Matlab dispone de una función denominada `chol( )` que realiza esta descomposición si es posible. Así, si escribimos

```
A=pascal(6);
R=chol(A)
```

obtenemos el resultado

```
R=
    1    1    1    1    1    1
    0    1    2    3    4    5
    0    0    1    3    6   10
    0    0    0    1    4   10
    0    0    0    0    1    5
    0    0    0    0    0    1
```

5.3. Métodos iterativos

Una posible implementación para Matlab del método de Jacobi viene dada en la siguiente función


```
function [x]=jacob(A,b)
% esta rutina implementa el metodo Jacobi
% basico.
%

tol=1.e-4;
itmax=1000;

[n,n]=size(matriz);
xo=zeros(n,1);
it=0;
error=1000.0;

while it<=itmax & error >tol
 it=it+1;
 D=diag(A);
 D1=inv(D);
 x=D1*(D-A)*xo+D1*b;
 verr=x'-xo;
 error=norm(verr)/norm(x);
 xo=x';
end
   x=xo;
   disp('el numero de iteraciones es')
   disp(it)
   disp('el error es')
   disp(error)
```

Por otro lado, el método de Gauss-Seidel se puede implementar como en la siguiente función

```
function [x]=gaussseidel(matriz,vector)
% esta rutina implementa el metodo de Gauss-Seidel
% basico.
%
tol=1.e-4;
itmax=1000;
[n,n]=size(matriz);
```

```
xo=zeros(n,1);
it=0;
error=1000.0;

% calculo de las matrices
D=diag(diag(matriz))
E=-(tril(matriz)-D)
F=-(triu(matriz)-D)

while it<=itmax & error >tol
   it=it+1
   x=(D-E)\(F*x0+vector);
   error=norm(x-x0)/ norm(x)
   x0=x;
end
   disp('el numero de iteraciones es')
   disp(it)
   disp('el error es')
   disp(error)
```

Por último una función que implementa el método SOR para un ω dado es la siguiente

```
 function [x]=sor1(matriz,vector,w);
[n,n]=size(matriz);
tol=1.e-4;
itmax=300;
x0=zeros(n,1);
it=0;
error=1000.0;
% calculo de las matrices
D=diag(diag(matriz))
E=-(tril(matriz)-D)
F=-(triu(matriz)-D)

while it<=itmax & error >tol
   it=it+1
   x=(D-w*E)\(w*F*x0+(1-w)*D*x0+w*vector);
   error=norm(x-x0)/ norm(x)
```



```
   x0=x;
end

   disp('el numero de iteraciones es')
   disp(it)

   disp('el error es')
   disp(error)
```

El Matlab dispone de funciones implementadas para la resolución de sistemas de ecuaciones lineales mediante métodos iterativos como `bicg( )`, `bicgstab` , `cgs( )`, `gmres( )` , `lsqr( )` , `pcg( )` , etc., pero se basan en métodos más complejos que los que hemos estudiado en la teoría y sobrepasan las pretensiones del curso.

5.4. Ejercicios

1. Una viga horizontal flexible empotrada en un extremo A y libre en el extremo B se considera que tiene cuatro grados de libertad traslacional $u_1, \ldots, u_4$, donde u_i se localiza a $i/5$ de la distancia de A a B. Si se aplica una carga unitaria en u_3, el vector $u = (u_1, u_2, u_3, u_4)^T$, satisface el sistema

$$Ku = r \ ,$$

donde la matriz de rigidez K y el vector de cargas r, vienen dadas por

$$\begin{pmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{pmatrix} \ , \quad r = \begin{pmatrix} 0 \\ 0 \\ EI \\ 0 \end{pmatrix} \ .$$

la constante EI depende del material de la viga y de su geometría. Calcular u cuando $EI = 1$.

2. La distribución de temperatura en el estado estacionario de una placa plana se puede aproximar en 9 puntos internos aplicando la ecuación de Laplace discretizada en cada punto. Si se mantienen los lados de una placa cuadrada a temperaturas

constantes de 0° C y 100° C, las temperaturas en los 9 puntos se pueden obtener como la solución del siguiente sistema

$$\begin{aligned} -4T_1 + T_2 + T_4 &= -100 \\ T_1 - 4T_2 + T_3 + T_5 &= -100 \\ T_2 - 4T_3 + T_6 &= 200 \\ T_1 - 4T_4 + T_5 + T_7 &= 0 \\ T_2 + T_4 - 4T_5 + T_6 + T_8 &= 0 \\ T_3 + T_5 - 4T_6 + T_9 &= -100 \\ T_4 - 4T_7 + T_8 &= 0 \\ T_5 + T_7 - 4T_8 + T_9 &= 0 \\ T_6 + T_8 - 4T_9 &= -100 \end{aligned}$$

Obtener la temperatura en los distintos puntos de la placa.

3. Dado el sistema

$$\begin{aligned} x_1 + x_2 &= 4 \\ 2x_1 + 2x_3 &= 4 \\ 3x_2 + 3x_3 &= 4 \end{aligned}$$

a) Resuelve el sistema mediante el método de Gauss sin pivotación. b) Resuelve el sistema mediante el método de Gauss con pivotación parcial.

4. Escribid una función de Matlab que implemente el algoritmo de Thomas para una matriz tridiagonal simétrica. Utilízala para obtener la solución del sistema

$$\begin{bmatrix} 4 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} .$$

5. Recuerda que una matriz simétrica y definida positiva admite factorización de Cholesky. Modifica la función `trisuper( )` para construir una nueva función que permita resolver un sistema de ecuaciones asociado a una matriz triangular inferior. Usar esta nueva función, la función `trisuper( )` y la función `chol( )`, para resolver el

siguiente sistema

$$\begin{bmatrix} 4 & -1 & 1 \\ -1 & 4{,}25 & 2{,}75 \\ 1 & 2{,}75 & 3{,}5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -3 \\ 15{,}75 \\ 17 \end{bmatrix}$$

6. Construye el sistema de ecuaciones que han de cumplir los coeficientes de un polinomio de tercer grado

$$y = c_3x^3 + c_2x^2 + c_1x + c_0 \ ,$$

para que pase por los puntos $(12, 8)$, $(3, 27)$, $(6, 64)$, $(5, 125)$. Utiliza la función `cond( )` para obtener el número de condición de la matriz del sistema. Obtén la solución del sistema y dibuja los puntos y el polinomio obtenido.

7. Construye una función de Matlab que implemente el método SSOR. Dado el sistema

$$\begin{aligned} 4x_1 + 3x_2 &= 24 \\ 3x_1 + 4x_2 - x_3 &= 30 \\ -x_2 + 4x_3 &= -24 \end{aligned}$$

compara tres iteraciones del método SOR y 3 iteraciones del método SSOR tomando $x_0 = (1, 1, 1)^T$ y $\omega = 1{,}25$. Compara estos resultados refiriéndolos a la solución obtenida con el operador .

Práctica 6

Interpolación y ajuste de los datos de una tabla

El paquete MatLab dispone de distintas funciones para la interpolación y ajuste de los datos de una tabla de simple entrada. Revisaremos estas funciones y estudiaremos algunos ejemplos de su funcionamiento.

6.1. Polinomios en Matlab

Para representar un polinomio Matlab utiliza un vector con los coeficientes del mismo ordenados en potencias de x decrecientes. Así, el vector

```
1 -4 -3 -4 10
```

representa el polinomio

$$P(x) = x^4 - 4x^3 - 3x^2 - 4x + 10 \ .$$

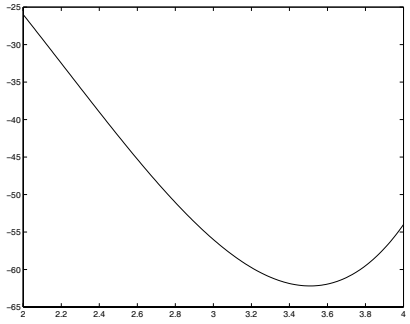
Si p es un vector que contiene los coeficientes de un polinomio en potencias decrecientes, podremos evaluarlo para un valor concreto de la variable s_0 con el comando `polyval(p,s0)`. Esta función nos permite obter una gráfica del polinomio en un cierto intervalo. Por ejemplo, si introducimos en Matlab las instrucciones

```
p=[1   -4  -3  -4  10];
x=2:0.01:4;
```



```
y=polyval(p,x);
plot(x,y)
```

obtenemos la gráfica del polinomio $P(x)$ en el intervalo $[2, 4]$, que es de la forma,



Las raíces de un polinomio se obtienen mediante la función `roots( )`. Así la instrucción

```
raices=roots(p)
```

da como resultado las raíces

```
4.7200
-0.8600+1.1743i
-0.8600-1.1743i
1.0000
```

Se puede reconstruir el polinomio a partir de sus raíces haciendo

```
p=poly(raices)
```

Supongamos que se tienen los polinomios

$$\begin{aligned} a(x) &= x^2 + 3x + 2 \ , \\ b(x) &= x^2 + 5x + 1 \ , \end{aligned}$$

escribiremos los vectores de coeficientes comenzando por el coeficiente que acompaña al mayor exponente e iremos colocando los coeficientes que acompañen a las potencias de x de forma decreciente, es decir,

```
a = [1 3 2]
b = [1 5 1]
```

el producto de polinomios se obtiene con la función `conv( )`, así,

```
conv(a,b)
```

da como resultado

```
 1 8 18 13 2
```

que corresponde al polinomio

$$x^4 + 8x^3 + 18x^2 + 13x + 2 \ .$$

6.2. Interpolación

Dada una tabla de datos de simple entrada

x_0	x_1	$\cdots$	$\cdots$	$\cdots$	x_n
y_0	y_1	$\cdots$	$\cdots$	$\cdots$	y_n

la función de Matlab `interp1( )` permite calcular interpolaciones para valores comprendidos entre los valores extremos recogidos en la tabla.

La sintaxis de la función es

```
yi=interp1(x,y,xi,'metodo')
```

donde `x` es el vector de las xs de la tabla cuyos valores deben estar ordenados de forma creciente o decreciente. `y` es el vector de las ys recogidas en la tabla. `xi` es el valor o valores para los que se quieren obtener las interpolaciones. `'metodo'` es la técnica utilizada para la interpolación. Este argumento es opcional y puede tomar los siguientes valores:

`'linear'` Se realiza una interpolación lineal.

`'spline'` Se realiza una interpolación construyendo un splin cúbico asociado a los datos de la tabla.

`'cubic'` Se realiza una interpolación cúbica entre los datos de la tabla. Para poder utilizar esta opción, los datos del vector `x` deben estar igualmente espaciados.

Veamos un ejemplo. En la siguinte tabla se muestra la población de Estados Unidos, en millones de personas desde 1900 hasta 1990.

año	1900	1910	1920	1930	1940
Pobl.	75.995	91.972	105.711	123.203	131.669
año	1950	1960	1970	1980	1990
Pobl.	150.697	179.323	203.212	226.505	249.633

Podemos escribir las instrucciones

```
tiem=1900:10:1990;
pob=[75.995   91.972 105.711 123.203 131.669 ...
     150.697 179.323 203.212 226.505 249.633];
```

con lo que tenemos dos vectores con los datos de la tabla. Hemos de resaltar que los tres puntos se usan en la definición de `pob` para poder cabiar de línea sin que Matlab interprete que ha finalizado la instrucción.

Podemos estimar la población de 1975 escribiendo

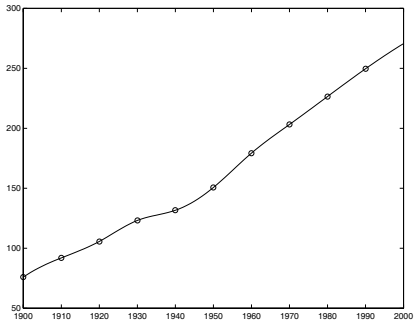
```
interp1(tiem,pob,1975)
```

obteniendo como resultado 214.8585 millones.

Por otra parte, podemos dibujar la evolución de la población mediante las instrucciones siguintes:

```
xi=1900:2000;
yi=interp1(tiem,pob,xi,'spline')
plot(tiem,pob,'o',xi,yi)
```

obteniendo la gráfica



6.3. Ajuste

La función `polyfit( )` nos proporciona un ajuste polinómico de los datos de una tabla. La sintaxis de la función es la siguiente:

```
p=polyfit(x,y,n)
```

donde `x` e `y` son vectores que contienen los datos de la tabla, `n` es el grado del polinomio que se quiere ajustar, y `p` es el vector de coeficientes del polinomio obtenido para el ajuste.

Veamos un ejemplo donde se puede utilizar esta función.

Se sabe que los datos de la siguiente tabla

x	1	2	3	4	5
y	4.52	4.09	3.70	3.35	3.03
x	6	7	8	9	10
y	2.74	2.48	2.25	2.03	1.84

corresponden a un decaimiento exponencial de la forma

$$y = a\exp(bx) \ .$$

Para estimar los valores de a y b a partir de los datos de la tabla procedemos del siguiente modo.


```
x=1:10;
y=[ 4.52,4.09,3.70,3.35,...
3.03,2.74,2.48,2.25,2.03,1.84];
y1=log(y);
p=polyfit(x,y1,1)
```

y obtenemos

```
-0.0999 1.6082
```

que son los parámetros de la recta

$$\ln(y) = bx + \ln(a) \ ,$$

con lo que tenemos $b = -0{,}0999$ y $a = 4{,}9938$.

6.4. Ejercicios

1. Calcular:

   a) Las raíces quintas de -1.

   b) Calcula $(x^4 + 3x^3 + 2x + 0{,}5)(x^6 + 3x + 0{,}3)$

   c) Las raíces reales de $x^6 - 3x^5 - x^4 + 7x^3 - 14x^2 + 10x - 12$.

2. Dada la tabla en donde se muestra la evolución del nitrógeno de mineralización

Tiempo (días)	Nmin (mg/kg)
7	9.466
14	8.211
27	15.590
41	17.615
55	20.215
83	21.734

   Obtener el tiempo necesario para que el nitrógeno de mineralización sea de 18.5 mg/Kg. Utilizad para ello un interpolante lineal y un spline, y comparar los resultados.

3. Se conoce que la relación existente entre el peso vivo de las larvas de la mariposa nocturna, W (g), y el oxígeno consumido por la larva, R en ml/h, es aproximadamente de la forma

$$R = bW^a \ .$$

Obtened los valores de a y b a partir de los datos de la siguiente tabla

W	R
0.017	0.154
0.087	0.296
0.174	0.363
1.11	0.531
1.74	2.23
4.09	3.58
5.45	3.52
5.96	2.40

4. Bajo ciertas condiciones se conoce que la evolución de una población con el tiempo se puede modelizar mediante una ecuación logística de la forma

$$P(t) = \frac{1000}{1 + Ce^{At}} \ .$$

Obtened C y A para la siguiente tabla de datos:

$P(t)$	200	400	650	850	950
t	0	1	2	3	4

5. Del calibrado de un motor se han obtenido los siguintes datos

Tiempo (milliseconds)	Distancia (cm)
0.00	0.0
1.40	1.0
2.37	2.0
3.30	3.0
3.37	4.0
4.11	5.0
5.42	6.0
5.71	7.0
6.39	8.0
7.26	9.0
7.82	10.0
8.67	11.0
9.12	12.0
9.66	13.0
10.70	14.0
11.23	15.0
11.25	16.0
12.47	17.0
12.79	18.0
13.20	19.0
14.12	20.0
14.83	21.0
15.83	22.0
16.04	23.0

Ajustad un polinomio de grado 4 que modelice la distancia en función del tiempo. Haced una gráfica con los datos de la tabla y el polinomio ajustado.

6. La densidad de partículas que emergen de una lámina fina formando un ángulo θ con la perpendicular a la placa, satisface la siguiente relación

$$\text{densidad} = \frac{k}{\text{sen}\left(\frac{\theta}{2}\right)^n} \ ,$$

donde n es un entero positivo, y k una constante. Deducir el valor de estas dos constantes a partir de los siguientes datos

θ	$\frac{\pi}{36}$	$\frac{\pi}{12}$	$\frac{\pi}{6}$	$\frac{2\pi}{9}$	$\frac{5\pi}{18}$	$\frac{5\pi}{12}$	$\frac{7\pi}{12}$
densidad	8 10^5	2 10^4	800	500	107	26	8

7. Interpolad la función

$$f(x) = \frac{1}{1+x^2} \ ,$$

de dos modos, primero un conjunto de puntos $x_n = -5 + n$, $y_n = f(x_n)$ con $n = 0, 1, 2, \ldots, 10$, y luego utilizando otro conjunto de puntos, $x_n = 5\cos(\pi n/10)$, $y_n = f(x_n)$ con $n = 0, 1, 2, \ldots, 10$. Utilizad un polinomio de grado 9 para la interpolación de la funcón. Realizad una gráfica con las dos interpolaciones y los valores de la función.

Práctica 7

Introducción a las ecuaciones en derivadas parciales

El MatLab dispone de un 'toolbox' completo para el estudio de las ecuaciones en derivadas parciales. Nosotros aquí sólo veremos algunos métodos sencillos para la integración de estas ecuaciones.

Supongamos que se quiere estudiar el problema de contorno siguiente,

$$\frac{\partial^2 u}{\partial x^2} = -2 \ , \quad x \in (0,1) \ , \tag{7.1}$$

con $x(0) = 0$ y $x(10) = 1$. Utilizamos una aproximación en diferencias finitas

$$\frac{\partial^2 u}{\partial x^2}(x_i) \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} \ ,$$

si utilizamos 11 nodos para la discretización x_0, x_1, $\ldots$, x_{10}, se tiene que $h = 1$ y la ecuación (7.1) se puede aproximar como

$$u_{i-1} - 2u_i + u_{i+1} = -2 \ .$$

Haciendo variar $i = 1, \ldots, 9$, se obtiene el sistema de ecuaciones

$$\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -2 \\ -3 \end{bmatrix} .$$

Para resolver este sistema utilizamos las instrucciones

```
v1=ones(1,8);
v2=-2*ones(1,9);
A=diag(v2)+diag(v1,-1)+diag(v1,1);
b=-2*ones(9,1);
b(9)=-3;
solu=A\b;
solu=[0;solu;1]
plot(solu,'-o');
```

obteniendo el resultado

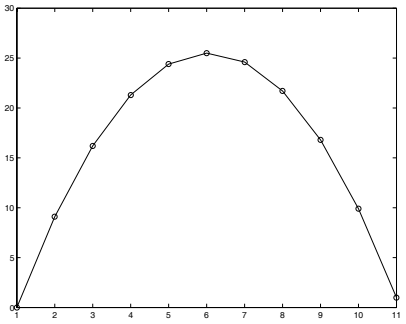


Figura 7.1.- Solución del problema de contorno.

Ahora consideraremos un problema dependiente del tiempo. Supongamos que se tiene una barra de longitud 0.5 m que inicialmente se halla a una temperatura de 25 grados centígrados. El extremo derecho se pone a una temperatura de 100 grados centígrados, mientras que el otro extremo se mantiene a 25 grados. Pretendemos saber cómo evoluciona la temperatura en los distintos puntos de la barra. Para ello, se usa la ley de Fourier

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \; ; \quad \alpha = 10^{-2}\,\mathrm{m}^2/\mathrm{s} \; .$$

Utilizamos el método explícito visto en teoría,

$$u_i^{n+1} = u_i^n + \frac{\alpha \Delta t}{\Delta x^2} \left(u_{i-1}^n - 2u_i^n + u_{i+1}^n \right) \; ,$$

y elegimos $\Delta x = 0{,}1$, y 6 nodos espaciales. Para que se cumpla la condición de Courant

$$0 < \frac{10^{-2}}{10^{-2}} \Delta t < 0{,}5 \; ,$$

tomamos, por ejemplo, $\Delta t = 0{,}1$.

Las siguientes instrucciones de MatLab resuelven el problema, mostrando la solución hasta 100 segundos despues de colocar la funete de calor. La soución se muestra cada 10 segundos y se generan 10 fiheros con las distintas configuraciones para la temperatura en la barra.

```
close all
deltat=0.1;
deltax=0.1;
alpha=1e-2;
u0=25*ones(6,1);
u0(6)=100;
it=1;
pasost=100;
nprint=10;
hold on
for n=1:pasost
    tiempo=n*deltat;
    u(1)=25;
    for i=2:5
        u(i)=u0(i)+alpha*deltat/(deltax^2)*...
            (u0(i-1)-2*u0(i)+u0(i+1));
```

```
    end
    u(6)=100;
  if(it==nprint)
      it=0;
      plot(u);
      fichero=['fig',num2str(tiempo),'.eps'];
      eval(['print ',fichero,' -deps']);
  end
  it=it+1;
  u0=u;
end
```

El formato elegido para generar los ficheros con las gáficas ha sido el 'encapsulated postscript'. Se puede consultar la ayuda de la función `print( )` para generar las figuras con otro formato.

7.1. Ejercicios

1. La distribución estacionaria de temperaturas en una varilla que se calienta, en una cierta aproximación, se puede escribir como el modelo en diferencias siguiente

$$x_{n+1} - 2x_n + x_{n-1} + 0{,}3 = 0 \ .$$

   Sabiendo que las temperaturas en los extremos son $x_0 = 1$ y $x_{20} = 10$. Obtén las temperaturas en los puntos intermedios, $x_1, x_2, \ldots, x_{19}$.

2. Resuelve el problema de la distribución de temperaturas dependiente del tiempo que se ha expuesto antes como ejemplo utilizando un paso de tiempo $\Delta t = 1$. Repite los cálculos con $\Delta t = 1$ pero usando el método implícito y el método de Crank-Nicholson expuestos en la teoría.

3. Un pulso viajero tiene la ecuación

$$y(x,t) = \exp\left(-20\,(x - 0{,}1t)^2\right) \ .$$

   crea 10 ficheros llamados `pulso1.jpg`, `pulso2.jpg`, ... , `pulso10.jpg` que tengan la figura de la evolución del pulso para los instantes $t = 1, 2, \ldots, 10$.

4. La deformación de una lámina rectangular y alargada que soporta una carga uniforme bajo una una fuerza de tensión axial se modeliza mediante una ecuación

diferencial de segundo orden. Denotemos por s el módulo de la fuerza axial y por q la intensidad de la carga uniforme, entonces la deformación w viene dada por

$$w'' - \frac{s}{D}w = -\frac{ql}{2D}x + \frac{q}{2D}x^2 \ ,$$

con $0 \le x \le l$, $w(0) = w(l) = 0$. Siendo l la longitud de la lámina y D su rigidez. Tomando $q = 200\text{kg/m}^2$, $s = 100\text{kg/m}$, $D = 8{,}8\,10^7\text{kg/m}$, $l = 50\text{m}$. Aproxima la deformación a intervalos de un metro y haz una gráfica con el resultado.