

MÉTODOS MATEMÁTICOS  
PARA LA INGENIERÍA QUÍMICA

PRÁCTICAS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA  
DEL DISEÑO  
DEPARTAMENTO DE MATEMÁTICA APLICADA  
UNIVERSIDAD POLITÉCNICA DE VALENCIA

## Índice general

<b>1. Introducción al programa MatLab</b>	<b>4</b>
1.1. Preguntas más frecuentes . . . . .	4
1.2. Operaciones elementales, variables y constantes . . . . .	6
1.3. Funciones . . . . .	7
1.4. Matrices. Operaciones con matrices . . . . .	8
1.5. Ejercicios . . . . .	11
<b>2. Bucles y funciones</b>	<b>13</b>
2.1. Bucles . . . . .	13
2.2. Ficheros.m . . . . .	16
2.3. Ejercicios . . . . .	17
<b>3. Gráficas con Matlab</b>	<b>20</b>
3.1. Gráficas 2D . . . . .	20
3.2. Gráficos 3D . . . . .	29
3.3. Ejercicios . . . . .	32
3.4. Ejercicios . . . . .	32
<b>4. Interpolación y ajuste de los datos de una tabla</b>	<b>34</b>
4.1. Polinomios en Matlab . . . . .	34
4.2. Interpolación . . . . .	36
4.3. Ajuste . . . . .	38

4.4. Ejercicios . . . . .	39
<b>5. Raíces de ecuaciones, integrales y ecuaciones diferenciales</b>	<b>43</b>
5.1. Búsqueda de raíces . . . . .	43
5.2. Integración aproximada . . . . .	46
5.3. Ecuaciones diferenciales . . . . .	46
5.4. Ejercicios . . . . .	47
<b>6. Sistemas dinámicos lineales</b>	<b>51</b>
6.1. Valores y vectores propios . . . . .	51
6.2. Ecuación característica . . . . .	52
6.3. Descomposición en fracciones simples de expresiones racionales . . . . .	53
6.3.1. Conversión de modelos . . . . .	55
6.4. Ejercicios . . . . .	59
<b>7. Ecuaciones en diferencias</b>	<b>64</b>
7.1. Problemas de valores iniciales . . . . .	64
7.2. Problemas de contorno . . . . .	67
7.3. Ejercicios . . . . .	69
<b>8. Cálculo simbólico, derivación e integración con MatLab. Aplicación a la variable compleja.</b>	<b>70</b>
8.1. Cálculo simbólico. . . . .	70
8.2. Resolución de ecuaciones . . . . .	74
8.3. Derivación simbólica con MatLab . . . . .	76
8.4. Integración simbólica con MatLab . . . . .	77
8.5. Ejercicios . . . . .	79

# 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 **Matrix Laboratory**.

Debido a su versatilidad 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 párrafo 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 desea 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( ) → logaritmo neperiano.
log10( ) → logaritmo 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  $1 \times 1$  son escalares, las matrices  $1 \times n$  ó  $n \times 1$  son vectores filas o columna, respectivamente

Para introducir los elementos 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  $3 \times 3$ .

```
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 múltiples 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,

$$A \setminus B \rightarrow \text{es equivalente a } A^{-1}B.$$

$$A / B \rightarrow \text{es equivalente a } AB^{-1}.$$

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 añadirles 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

$$B = [\text{zeros}(2,3), \text{eye}(2); \text{zeros}(3,2), \text{eye}(3)]$$

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

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

3. Calcular

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

4. 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)  $A \setminus b$  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  $b \wedge b$ .

5. 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 partir de las matrices  $D$  y  $E$  anteriores sacar sus vectores fila, sus vectores columna y una submatriz  $2 \times 2$  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

$$\left[ D \mid E \right], \left[ \frac{D}{E} \right].$$

6. Dada la matriz

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

obtener su parte estrictamente triangular inferior (sin la diagonal) y su parte estrictamente 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:

```
<, >, <=, >=, ==,
~ = significa no igual,
& significa y
~ 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

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



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

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

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

La instrucción

```
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 automóvil, 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  $B = \text{permuta}(A, k, i)$ .

9. Calcula aproximadamente el valor de  $\pi$  utilizando el siguiente resultado

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

¿ Cuántos términos de la serie hace falta sumar para obtener una precisión de  $10^{-12}$ ?

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

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

con  $F_0 = F_1 = 1$

- Calcula los 10 primeros números de Fibonacci.
- 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 resultados?

## 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.

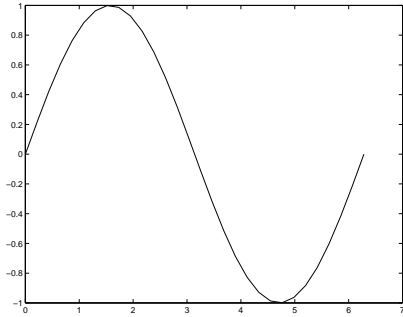
#### 3.1. Gráficas 2D

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 abscisas los elementos del vector  $x$  y como ordenadas los elementos del vector  $y$ . Con el comando `plot(y)` toma como abscisas los números naturales  $1, 2, \dots, n$ . 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)
```

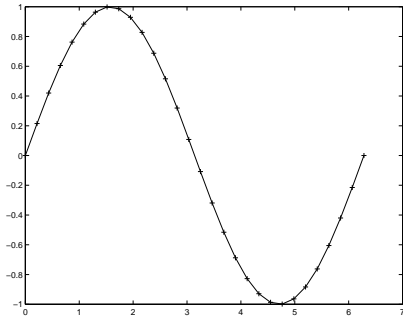
obtenemos la siguiente gráfica



Es posible generar la misma gráfica con líneas y cruces escribiendo

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

con lo que se obtiene:



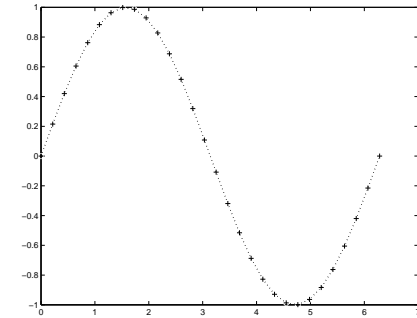
Por otra parte, se pueden controlar los colores y los estilos de las líneas utilizando los símbolos de la siguiente tabla:

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+')
```

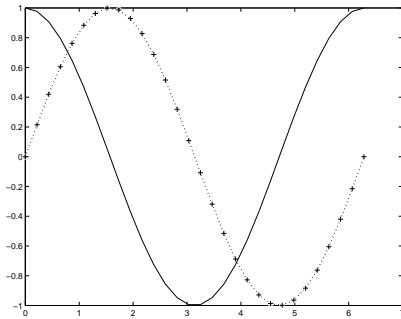
obteniéndose



Cada vez que se ejecuta el comando `plot` desaparece la figura anterior. Si pretendemos superponer gráficas el comando `hold on` 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)
```

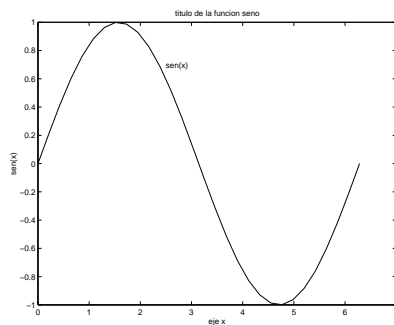
se obtiene la gráfica siguiente:



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 funciones `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('título 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.

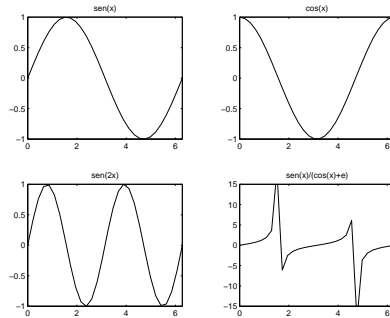
`axis off`: no dibuja los ejes.

`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 selecciona la segunda zona
plot(x,z)
axis([0, 2*pi, -1, 1])
title('cos(x)')
subplot(2,2,3) % se selecciona la tercera zona
plot(x,a)
axis([0, 2*pi, -1, 1])
title('sen(2x)')
```

```
subplot(2,2,4) % se selecciona 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,

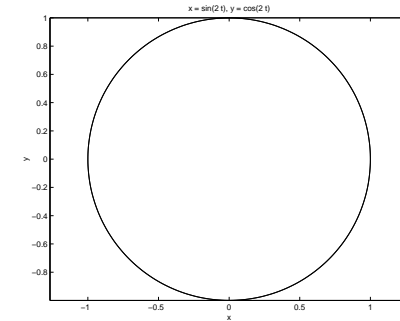
**loglog** : Es una función igual que **plot** pero usa escalas logarítmicas para el eje  $x$  y el eje  $y$ .

**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  $x$ .

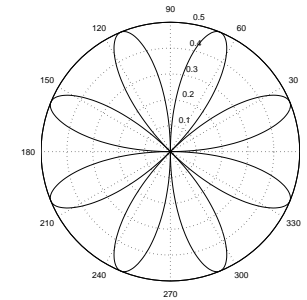
Si pretendemos dibujar curvas que vienen representadas en coordenadas paramétricas podremos utilizar el comando **ezplot** la instrucción sería como sigue:

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



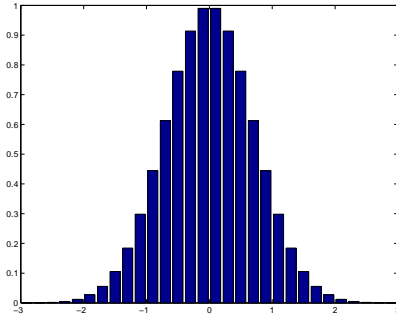
Una gráfica de una función en coordenadas polares puede crearse usando la función **polar**, como muestra el siguiente ejemplo:

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



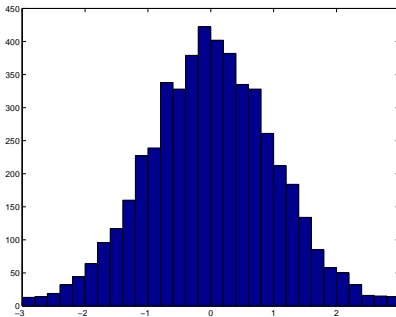
La función **bar** genera un diagrama de barras. Un ejemplo del diagrama de barras asociado a una gaussiana es el siguiente:

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



La función `hist` genera el histograma 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)
```

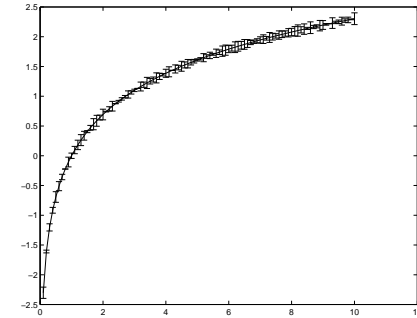


¿Qué pasa si se sustituye la función `randn` por la función `rand`?

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);
```

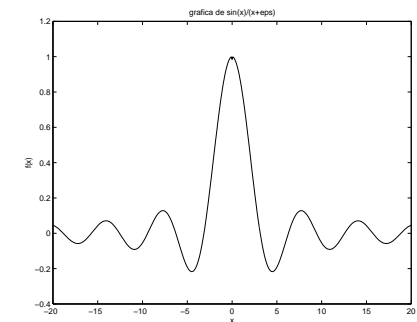
```
e=rand(size(x))/10; % crea un vector de errores aleatorio
errorbar(x,y,e)
```



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)')
```



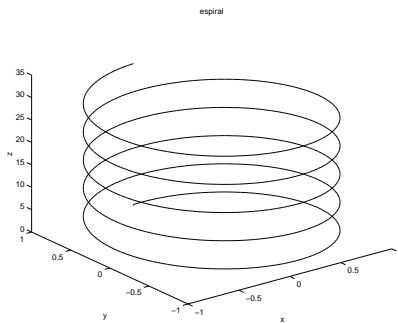
## 3.2. Gráficos 3D

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

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

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')
```



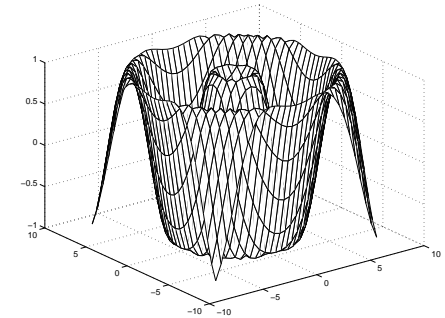
Se puede controlar la escala de los ejes mediante la función `axis([xmin, xmax, ymin, ymax, zmin, zmax])` de forma similar a como se hace para los gráficos bidimensionales. También la función `text(x,y,z,'texto')` nos permite escribir un texto en las coordenadas  $(x, y, z)$  del gráfico.

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

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

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 dirección 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

### 3.4. Ejercicios

1. Representar gráficamente las siguientes curvas:

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

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

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

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

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:

$$y = \tan(x), \quad x \in [0, 0,5],$$

$$y = \cosh(5 * x), \quad x \in [0, 10],$$

$$y = e^{2x} + 5 \text{sen}(2x), \quad x \in [0, \pi],$$

$$y = x^2 + 3x + 4, \quad x \in [0, 1].$$

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

$$y = 2x^3 + 5,4x^2 + 4,8x + 1,4,$$

$$y = 7x + 10.$$

5. Obtener **dos** representaciones 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

### 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.

#### 4.1. 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

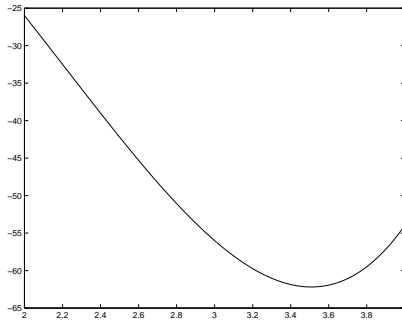
$$P(x) = x^4 - 4x^3 - 3x^2 - 4x + 10 .$$

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

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

## 4.2. Interpolación

Dada una tabla de datos de simple entrada

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

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  $x$ s de la tabla cuyos valores deben estar ordenados de forma creciente o decreciente.  $y$  es el vector de las  $y$ s 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 spline 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 siguiente 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

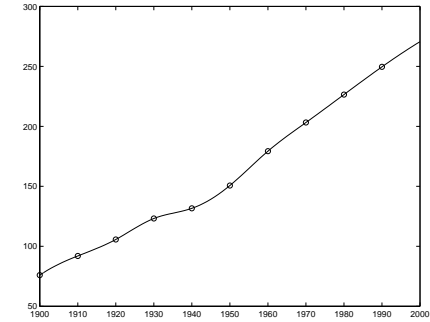
```
interp1(tiem,pob,1975)
```

obteniendo como resultado 214.8585 millones.

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

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

obteniendo la gráfica



### 4.3. 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

<i>x</i>	1	2	3	4	5
<i>y</i>	4.52	4.09	3.70	3.35	3.03
<i>x</i>	6	7	8	9	10
<i>y</i>	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  $a = 4,9938$ .

#### 4.4. Ejercicios

- Calcular:
  - Las raíces quintas de -1.
  - Calcula  $(x^4 + 3x^3 + 2x + 0,5)(x^6 + 3x + 0,3)$  en el punto  $x = 3$ .
  - Las raíces reales de  $x^6 - 3x^5 - x^4 + 7x^3 - 14x^2 + 10x - 12$ .
- 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.

- 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

- 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

- Del calibrado de un motor se han obtenido los siguientes 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. Es ampliamente aceptado que la tasa de crecimiento de bacterias respecto de tiempo es proporcional al número de bacterias presente en el cultivo. El factor de proporcionalidad,  $C$ , no es constante y está relacionado con unas constantes  $M$  y  $k$ , que dependen del tipo de bacterias, y de  $q$ , la cantidad de comida, del modo siguiente

$$\frac{1}{C} = \frac{k}{M} \frac{1}{q} + \frac{1}{M} .$$

Haz una estimación de los valores de estas constantes  $M$  y  $k$  a partir de los datos de la tabla siguiente

$q$	7	9	15	25	49	75	150
$C$	0.29	0.37	0.48	0.65	0.80	0.97	1.07

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

## Práctica 5

# Raíces de ecuaciones, integrales y ecuaciones diferenciales

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

### 5.1. Búsqueda de raíces

Supongamos que se buscan raíces de la función

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

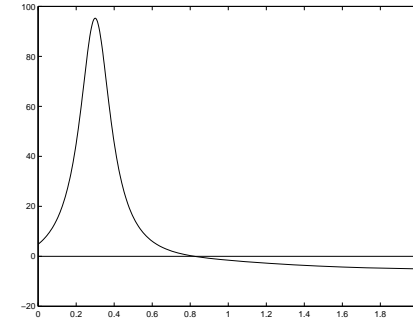
Primero se dibuja la función junto con el eje de las  $x$ -s, para ello, primero se define la función

```
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

$$\begin{aligned} x_1^2 - 10x_1 + x_2^2 + 8 &= 0, \\ x_1x_2^2 + x_1 - 10x_2 + 8 &= 0. \end{aligned} \quad (5.1)$$

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 (5.1) 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 jacobiana 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.

## 5.2. 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)
```

## 5.3. 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.

## 5.4. 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,

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

47

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. La concentración de una bacteria contaminante en un lago decrece según la expresión

$$c(t) = 80e^{-2t} + 20e^{-0.5t},$$

siendo  $t$  el tiempo medido en horas. Determinar el tiempo que se necesita para que la concentración del número de bacterias se reduzca a 7.

4. La temperatura de un cierto gas en grados Kelvin, varía con el tiempo según la expresión

$$T(t) = 400 + 200 \cos\left(\frac{2\pi t}{1440}\right),$$

mientras que su presión en atmósferas es de la forma

$$P(t) = \exp\left(-\frac{t}{1440}\right).$$

Calcular para qué valor del tiempo el volumen molar es de 50 litros si  $R = 0,082051 \cdot \text{atm}/\text{K}$ . El volumen se puede calcular utilizando la ley de los gases perfectos

$$V = \frac{RT}{P}.$$

5. 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

48



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

$$W = \int_{32.5}^{80} P dV .$$

6. 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  $2\pi$  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 utilizando Matlab.

7. Calcular aproximadamente el valor de

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

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

8. Se quiere saber la cantidad de calor que se necesita para aumentar la temperatura de una masa  $m = 200\text{g}$  de la temperatura  $T_1 = -100^\circ\text{C}$  a  $T_2 = 200^\circ\text{C}$ , si sabemos que

$$\text{cantidad de calor} = m \int_{T_1}^{T_2} c(T) dT ,$$

y tenemos la siguiente tabla de temperaturas y capacidades caloríficas

$T$	-100	-50	0	50	100	150	200
$c$	0.11912	0.12427	0.13200	0.13987	0.15032	0.15544	0.17408

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

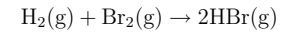
$$L \frac{dI}{dt} + RI + \frac{1}{C}Q = E(t) .$$

Supongamos que se tiene una resistencia de  $2\Omega$ , un condensador de 1 F y una autoinducción de 0.5 H. Si la fuerza electromotriz del circuito es de la forma  $E(t) = 5 \sin(t)$ . 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.

10. Supongamos que en un reactor tiene lugar la reacción



cuya ley de velocidades es de la forma

$$\frac{1}{2} \frac{d[\text{HBr}]}{dt} = [\text{H}_2][\text{Br}_2]^{\frac{1}{2}} ,$$

Suponiendo que la evolución de la concentración de  $\text{Br}_2$  en el reactor sigue una ley de la forma

$$[\text{Br}_2] = 5 \ln(1 + t) ,$$

y que se mantiene la concentración de hidrógeno constante  $[\text{H}_2] = 5$ . Si inicialmente no hay HBr, calcular la concentración de HBr a los 5 segundos de iniciada la reacción.

11. La población de una especie sigue la siguiente ley

$$y' = -0.5y - 0.04y^2 ,$$

donde  $t$  se mide en días e  $y(t)$  en millones. Sabiendo que hoy la población de esa especie es de 30 millones:

- ¿Cuántos individuos habrá pasada una semana?.
- ¿Cuándo se alcanzarán los 5 millones de individuos?.

## Práctica 6

### Sistemas dinámicos lineales

El Matlab dispone de ciertas funciones que permiten analizar el comportamiento de sistemas dinámicos y transformar estos modelos en las diferentes representaciones que suelen aparecer en los procesos que involucran problemas de control.

Antes de plantearnos este tipo de problemas vamos a estudiar algunos comandos que nos serán de utilidad.

#### 6.1. Valores y vectores propios

El comando que nos permite encontrar los valores propios y vectores propios de una matriz cuadrada  $A$  es `eig(A)`.

Si queremos obtener los valores propios y los vectores propios de una matrix  $A$  debemos realizar la siguiente asignación

$$[X,D]=\text{eig}(A)$$

como resultado se obtienen dos matrices,  $X$  y  $D$ . En las columnas de  $X$  se tienen los vectores propios correspondientes a los valores propios que aparecen en la diagonal de la matriz  $D$ .

**Ejemplo:** Sea

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

Después de asignar la matriz  $A$ , para obtener sus valores y vectores propios ejecutamos el comando

$$[X,D]=\text{eig}(A)$$

obteniendo

$X =$

$$\begin{bmatrix} 0.6993 & 0.3698+0.6093i & 0.3698-0.6093i & -0.0000 \\ 0.4203 & -0.3769-0.2368i & -0.3769+0.2368i & -0.5774 \\ 0.5426 & 0.1717-0.4863i & 0.1717+0.4863i & -0.5774 \\ 0.1995 & -0.1111+0.1244i & -0.1111-0.1244i & 0.5774 \end{bmatrix}$$

$D =$

$$\begin{bmatrix} 4.72000 & 0 & 0 & \\ 0 & -0.8600+1.1743i & 0 & 0 \\ 0 & 0 & -0.8600-1.1743i & 0 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

#### 6.2. Ecuación característica

Las raíces de la ecuación característica de una matriz  $A$  coinciden con los valores propios de  $A$ . Los coeficientes del polinomio característico de  $A$  se obtienen con el comando `poly(A)`. Así utilizando, para el ejemplo anterior, el comando

$$p=\text{poly}(A)$$

obtenemos

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

indicándonos que el polinomio característico de  $A$  es

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

Las raíces del polinomio característico se obtienen mediante la función `roots()`. Así la función

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

da como resultado

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

que coinciden con los valores propios de  $A$ .

Se puede reconstruir el polinomio a partir de sus raíces haciendo

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

### 6.3. Descomposición en fracciones simples de expresiones racionales

Si consideramos una función de racional

$$G(s) = \frac{b_1s^n + b_2s^{n-1} + \dots + b_{n-1}s + b_n}{a_1s^n + a_2s^{n-1} + \dots + a_{n-1}s + a_n}, \quad a_1 \neq 0.$$

su numerador y denominador los podemos representar mediante los vectores

$$\begin{aligned} num &= [b_1, b_2, \dots, b_n], \\ den &= [a_1, a_2, \dots, a_n]. \end{aligned}$$

El comando

```
[r,p,k]=residue(num,den)
```

nos proporciona los datos de la descomposición en fracciones simples, donde en el vector  $\mathbf{r}$  tenemos los numeradores de las fracciones, en el vector  $\mathbf{p}$  los polos del sistema y en  $\mathbf{k}$  el valor de la división exacta.

Si las raíces del denominador o polos son de multiplicidad mayor que uno, deberemos tener cuidado a la hora de interpretar la respuesta de Matlab ya que la descomposición incluirá términos de la forma

$$\frac{r_j}{s - p_j} + \frac{r_{j+1}}{(s - p_j)^2} + \dots + \frac{r_{j+n-1}}{(s - p_j)^n},$$

siendo  $n$  la multiplicidad de la raíz  $p_j$ .

El proceso contrario, es decir, dada una descomposición en fracciones simples obtener una expresión racional adecuada se realizará con el comando

```
[num,den]=residue(r,p,k)
```

**Ejemplo:** Dada la función racional

$$G(s) = \frac{2s^3 + 5s^2 + 3s + 6}{s^3 + 6s^2 + 11s + 6}$$

asignamos el numerador y el denominador y aplicamos el comando anterior

```
num=[2 5 3 6];
den=[1 6 11 6];
[r p k]=residue(num,den)
```

obteniendo el resultado

```
r =
-6.0000
-4.0000
3.0000
```

```
p =
-3.0000
-2.0000
-1.0000
```

```
k =
2.0000
```

Como se puede observar todas las raíces son simples y, por tanto, la descomposición es de la forma

$$G(s) = -\frac{6}{s+3} - \frac{4}{s+2} + \frac{3}{s+1} + 2.$$

Si aplicamos el comando al contrario, se tiene

```
[num,den]=residue(r,p,k)
```

```
num =
2.0000 5.0000 3.0000 6.0000
```

```
den =
1.0000 6.0000 11.0000 6.0000
```

Consideremos ahora la función racional

$$F(s) = \frac{s+2}{s^3+2s^2+s}$$

aplicando el comando `residue()` tenemos

```
num=[1 2];
den=[1 2 1 0];
[r,p,k]=residue(num,den)
```

obteniendo los resultados

$$r = \begin{bmatrix} -2 \\ -1 \\ 2 \end{bmatrix}, \quad p = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}, \quad k = []$$

Se puede observar que la raíz  $s = -1$  es doble y por lo tanto, el resultado se interpretará como sigue

$$F(s) = \frac{-2}{s+1} + \frac{-1}{(s+1)^2} + \frac{2}{s}.$$

### 6.3.1. Conversión de modelos

El Matlab dispone de funciones que nos permiten pasar de un sistema dinámico representado en su modelo espacio-estado a un sistema representado por su función o matriz

de transferencia. Matlab para este fin dispone de la función `tf2ss()`. Veamos cómo funciona esta función.

Se parte de un sistema de la forma

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ \dot{y} &= Cx + Du, \end{aligned}$$

y se llega a una expresión de la forma

$$\frac{Y_j(s)}{U_i(s)} = C(sI - A)^{-1}B + D.$$

La sintaxis de la función es la siguiente:

```
[num, den]= ss2tf(A,B,C,D,iu)
```

donde `iu` indica la coordenada asociada a cada entrada.

**Ejemplo.-** Dado el sistema

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -25 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \end{aligned}$$

introduciremos

```
A=[0,1;-25,4];
B=[1,1;0,1];
C=[1,0;0,1];
D=[0,0;0,0];
[num,den]=ss2tf(A,B,C,D,1)
```

obteniendo

```
NUM=
0 1 -4
0 0 -25

den=
1 -4 25
```

lo cual significa que

$$\frac{Y_1(s)}{U_1(s)} = \frac{s-4}{s^2-4s+25}, \quad \frac{Y_2(s)}{U_1(s)} = \frac{-25}{s^2-4s+25}.$$

Si introducimos

`[num,den]=ss2tf(A,B,C,D,2)`

obtenemos

NUM=

0 1 -3  
0 1 -25

den=

1 -4 25

con lo cual

$$\frac{Y_1(s)}{U_2(s)} = \frac{s-3}{s^2-4s+25}, \quad \frac{Y_2(s)}{U_2(s)} = \frac{s-25}{s^2-4s+25},$$

llegando pues a que la matriz de transferencia del sistema es de la forma

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} \frac{s-4}{s^2-4s+25} & \frac{s-3}{s^2-4s+25} \\ \frac{-25}{s^2-4s+25} & \frac{s-25}{s^2-4s+25} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$$

Por otro lado, la función `tf2ss( )` nos permite obtener una representación espacio-estado de un sistema dinámico a partir de su función de transferencia. Hay que tener en cuenta que esta representación no va a ser única.

**Ejemplo.-** Dado el sistema dinámico cuya función de transferencia es

$$\frac{Y(s)}{U(s)} = \frac{s}{s^3 + 14s^2 + 56s + 160}$$

podemos introducir

`num=[0 0 1 0];`

`den=[1 14 56 160];`

`[A,B,C,D]=tf2ss(num,den)`

obteniendo como resultado

A=

-14 -56 -160  
1 0 0  
0 1 0

B=

1  
0  
0

C=

0 1 0

D=

0

lo que implica que una posible representación espacio-estado del sistema es

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -14 & -56 & -160 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + [0]u$$

## 6.4. Ejercicios

1. Un modelo matemático de la fermentación de la bacteria *Pseudomonas ovalis*, que produce ácido glucónico es de la siguiente forma

$$\begin{aligned} \frac{dy_1}{dt} &= b_1 y_1 \left( 1 - \frac{b_1}{y_1} \right), \\ \frac{dy_2}{dt} &= \frac{b_3 y_1 y_4}{b_4 + y_4} - 0,9802 b_5 y_2, \\ \frac{dy_3}{dt} &= b_5 y_2, \\ \frac{dy_4}{dt} &= -1,011 \left( \frac{b_3 y_1 y_4}{b_4 + y_4} \right). \end{aligned} \quad (6.1)$$

donde,

$y_1$  = Concentración de células.

$y_2$  = Concentración de gluconolactona.

$y_3$  = Concentración de ácido glucónico.

$y_4$  = Concentración de glucosa.

las constantes  $b_1, \dots, b_5$  son parámetros del modelo, que dependen de la temperatura y el pH. Para las condiciones experimentales de 30°C y pH=6.6 los valores de los parámetros son

$b_1$	$b_2$	$b_3$	$b_4$	$b_5$
0.949	3.439	18.720	37.510	1.169

Obtener el estado estacionario del sistema.

Dado un sistema dinámico

$$\dot{\vec{x}} = \vec{F}(\vec{x}),$$

se puede linealizar alrededor del estado estacionario,  $\vec{x}^0$ , como

$$(\Delta \vec{x})' = D\vec{F}(\vec{x}^0) \Delta \vec{x},$$

donde  $\Delta \vec{x}$  es una perturbación y  $D\vec{F}(\vec{x}^0)$  es la matriz derivada de  $\vec{F}$  en el punto  $\vec{x}^0$ . Se pide, linealizar el sistema (6.1) alrededor del estacionario y discutir su estabilidad.

2. Dadas las funciones de transferencia

$$\begin{aligned} \frac{Y(s)}{U_1(s)} &= \frac{2s + 4}{s^3 + 1,3s^2 + 7s + 4} \\ \frac{Y(s)}{U_2(s)} &= \frac{s^3 + 4s + 2}{s^3 + 1,3s^2 + 7s + 4} \end{aligned}$$

Descomponedlas en fracciones simples y ver si el sistema dinámico asociado es o no estable.

3. De un sistema dinámico se sabe que su función de transferencia es de la forma

$$H(s) = \frac{1}{s^2 + 0,9s - 0,1}.$$

Discutid la estabilidad del sistema. Obtened unas ecuaciones para el sistema en el dominio temporal y estudiad la respuesta del sistema a una entrada de la forma  $u(t) = \text{sen}(t)$ , suponiendo condiciones iniciales nulas.

4. Obtened la matriz de transferencia del sistema espacio-estado

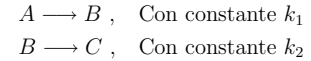
$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

donde

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -100 & -80 & -32 & -8 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 5 \\ 60 \end{bmatrix}, \quad C = [1 \ 0 \ 0 \ 0], \quad D = 0.$$

5. Se tienen dos reacciones químicas consecutivas



de las que se conoce que la evolución en el tiempo de la concentración de la especie  $B$  viene dada por los valores

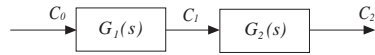
$t$	0	0.5	1	1.5	2	2.5	3	5	10
$[B]$	0	1.003	1.500	1.725	1.815	1.820	1.782	1.505	0.892

La evolución de las concentraciones de  $A$  y de  $B$  en función del tiempo vienen dadas por las ecuaciones

$$\begin{aligned} \frac{d[A]}{dt} &= -k_1[A] , \\ \frac{d[B]}{dt} &= -k_2[B] + k_1[A] , \\ [A](0) &= [A](t) + [B](t) + [C](t) . \end{aligned} \quad (6.2)$$

Del sistema se conoce que  $k_1 \in [0, 1]$  y  $k_2 \in [0, 0,5]$ , y que inicialmente se tiene una concentración de  $[A](0) = 8$  moles. Se pide:

- i) Estimar los valores de  $k_1$  y  $k_2$  que ajusten los datos experimentales.
  - ii) Obtener la evolución en el tiempo de las concentraciones de las distintas especies que intervienen en las reacciones, con estos datos.
  - iii) Si al cabo de 3 seg. de iniciada la reacción, se añaden 2 moles de de  $B$ , ¿Como sería la evolución de las concentraciones de  $A$ ,  $B$  y  $C$  a partir de ese tiempo?.
  - iv) Escribir el sistema (6.2) en forma matricial, obtener los autovalores de la matriz del sistema y discutir la estabilidad del mismo.
6. Un proceso químico tiene lugar mediante dos reactores en serie en dos etapas según muestra la figura

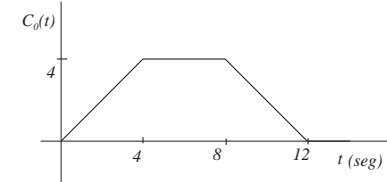


donde

$$G_1(s) = \frac{1}{\tau_1(s + a_1)} , \quad G_2(s) = \frac{1}{\tau_2(s + a_2)} ,$$

siendo  $a_i = \frac{k_i+1}{\tau_i}$  Si  $\tau_1 = 8$  min.,  $\tau_2 = 10$  min.,  $k_1 = 0,01$  min<sup>-1</sup> y  $k_2 = 0,09$  min<sup>-1</sup> se pide:

- i) Obtener la función de transferencia del sistema global.
- ii) Obtener la ecuación diferencial asociada al sistema.
- iii) Estudiar la evolución de  $C_2(t)$  sabiendo que la entrada  $C_0(t)$  es de la forma



7. Dado el sistema

$$\begin{aligned} \dot{x}_1 &= -x_1 - x_2 + u_1 + u_2 \\ \dot{x}_2 &= 6,5x_1 + u_1 \\ y_1 &= x_1 \\ y_2 &= x_2 \end{aligned}$$

Obtener su matriz de transferencia y discutir su estabilidad. Utilizar la función `step()` de Matlab para estudiar la respuesta del sistema a entradas dadas por funciones escalón.

8. De un sistema dinámico se sabe que su matriz de transferencia es de la forma

$$G(s) = \begin{bmatrix} \frac{s+4}{s^2+4s+25} \\ \frac{-25}{s^2+4s+25} \end{bmatrix}$$

Utilizar la función `tf2ss()` de Matlab para obtener una representación espacio-estado del sistema. Obtener la evolución de las salidas del sistema si se tiene una entrada de la forma

$$u(t) = \text{sen}(\pi t) + \text{cos}(\pi t)$$

para  $t \in [0, 10]$ .

9. Se tiene una reacción de primer orden que tiene lugar en una batería de 4 reactores dispuestos en serie. La reacción tiene una constante de velocidad  $k = 0,097$  min<sup>-1</sup> y un tiempo de residencia de  $\tau = 8$  min. La función de transferencia de los reactores es de la forma

$$G_i(s) = \frac{1}{\tau(s + a)} , \quad i = 1, 2, 3, 4,$$

donde  $a = \frac{k+1}{\tau}$ .

Si se mantiene la concentración de entrada al primer reactor constante  $[C_1] = 10$ , estimar el estado estacionario para la concentración del cuarto reactor y el tiempo necesario para alcanzar dicho estacionario.

Diseñar distintos controles para el sistema de forma que el tiempo que tarde el sistema en alcanzar el estado estacionario se haga la mitad y el doble.

## Práctica 7

### Ecuaciones en diferencias

En esta práctica utilizaremos el Matlab para estudiar la solución de algunos problemas asociados con ecuaciones y sistemas de ecuaciones en diferencias.

#### 7.1. Problemas de valores iniciales

La evolución del número de individuos de algunas especies se puede modelizar para las distintas generaciones mediante una ecuación en diferencias, denominada *ecuación logística*, y que es de la forma

$$x_{n+1} = bx_n(1 - x_n) , \quad (7.1)$$

donde  $b$  es un parámetro que puede tomar distintos valores, y  $x_n$  es una función que describe el número de individuos en cada generación, normalizado de tal manera que  $x_n \in [0, 1]$ .

Trataremos de estudiar utilizando el Matlab el comportamiento de la solución de la ecuación (7.1) para distintos valores del parámetro  $b$ .

En primer lugar escribimos una función de Matlab de la forma

```
function x1=logistica(x0,b)
% Funcion logistica
x1=b*x0*(1-x0);
```

Suponemos una población inicial de  $x_0 = 0,5$ , fijamos  $b = 2,9$  y estudiamos la evolución de la población para 150 generaciones. Para ello, se utilizan las instrucciones

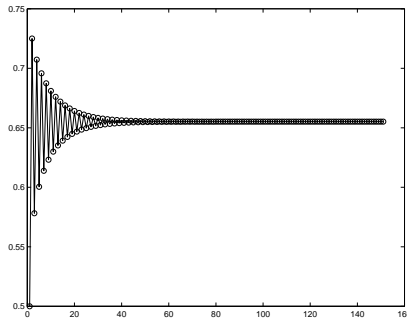


```

b=2.9;
x0=0.5;
x=[x0];
for i=1:150
    x=[x,logistica(x(i),b)];
end
plot(x, '-o');

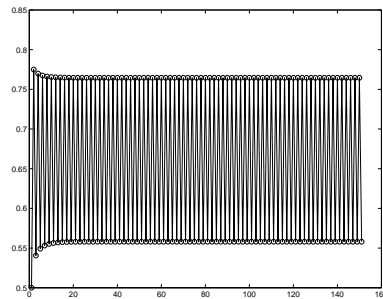
```

obteniendo como resultado



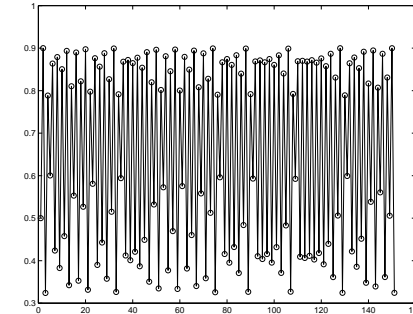
**Figura 7.1.-** Evolución de la ecuación logística.  $b = 2,9$ .

Si repetimos el proceso con  $b = 3,1$  se obtiene el siguiente comportamiento



**Figura 7.2.-** Evolución de la ecuación logística.  $b = 3,1$ .

observándose que la población tiene dos estados, y por tanto la solución es periódica de periodo 2. Si tomamos  $b = 3,6$  obtenemos la siguiente evolución



**Figura 7.3.-** Evolución de la ecuación logística.  $b = 3,6$ .

así, la solución obtenida ya no tiene un patrón reconocible y se dice que el sistema ha entrado en el régimen caótico.

Pasaremos a estudiar ahora un problema de valores iniciales asociado a una ecuación en diferencias de orden superior como el siguiente

$$x_{n+2} - 2x_{n+1} + x_n + (0,2\pi)^2 x_{n+1} = 0, \quad x_0 = 1, \quad x_1 = 1. \quad (7.2)$$

Este problema se puede abordar reabjando el orden de la ecuación de forma similar a como se hace en las ecuaciones diferenciales. Para ello, introducimos una función auxiliar

$$x_{n+1} = y_n,$$

y, por tanto, el problema (7.2) es equivalente al sistema

$$\begin{aligned} x_{n+1} &= y_n \\ y_{n+1} &= (2 - (0,2\pi)^2)y_n - x_n \end{aligned}$$

con las condiciones  $x_0 = 1, y_0 = 1$ . Para estudiar la evolución del sistema podemos utilizar las siguientes instrucciones de Matlab

```

x0=1;
y0=1;

```

```

x=[x0];
y=[y0];
for i=1:30
    x=[x,y(i)];
    y=[y,(2-(0.2*pi)^2)*y(i)-x(i)];
end
plot(x,'-o');

```

obteniendo la gráfica

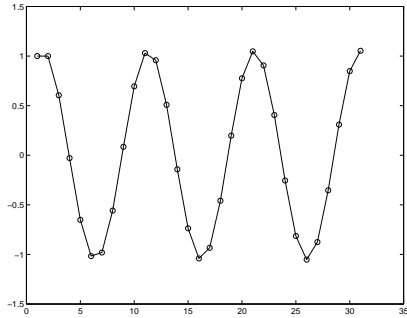


Figura 7.4.- Evolución de  $x_n$ .

## 7.2. Problemas de contorno

Otro problema relacionado con las ecuaciones en diferencias de orden superior, son los problemas de contorno. Por ejemplo, supongamos que nos dan la ecuación

$$x_{n+2} - (2 - (0,2\pi)^2)x_{n+1} + x_n = 0 ,$$

con las condiciones  $x_0 = 0$  y  $x_{10} = 1$ , y nos piden calcular  $x_1, x_2, \dots, x_9$ . Para resolver este problema planteamos las ecuaciones resultantes para  $n = 0, 1, \dots, 8$ .

$$\begin{aligned}
 x_0 - (2 - (0,2\pi)^2)x_1 + x_2 &= 0 \\
 x_1 - (2 - (0,2\pi)^2)x_2 + x_3 &= 0 \\
 &\vdots \\
 x_8 - (2 - (0,2\pi)^2)x_9 + x_{10} &= 0 .
 \end{aligned}
 \tag{7.3}$$

Llamando  $\alpha = (2 - (0,2\pi)^2)$  y teniendo en cuenta las condiciones  $x_0 = 0$  y  $x_{10} = 1$ , podemos escribir las ecuaciones (7.3) matricialmente de la forma

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

Para resolver este sistema utilizamos las instrucciones

```

v1=ones(1,8);
v2=-(2-(0.2*pi)^2)*ones(1,9);
A=diag(v2)+diag(v1,-1)+diag(v1,1);
b=zeros(9,1);
b(9)=-1;
solu=A\b;
solu=[0;solu;1]
plot(sol,u,'-o');

```

obteniendo el resultado

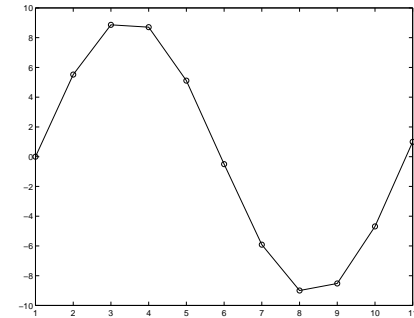


Figura 7.5.- Solución del problema de contorno.

### 7.3. Ejercicios

1. Dado el problema de valores iniciales

$$x_n + \sum_{j=1}^9 (0,5 + j)x_{n-j} = 0 ,$$

con  $x_i = i$ ,  $i = 0, \dots, 8$ . Calcula una expresión analítica para la solución del problema.

2. Obtén el valor de  $b$  en la ecuación logística para que se obtenga como estacionario una población con 4 estados.
3. Analiza la estabilidad de la ecuación

$$x_{n+2} + (\alpha - 1)x_{n+1} + 0,25(\alpha^2 - 2\alpha + 1)x_n = 0 ,$$

en función de los valores de  $\alpha$ . Toma un valor de  $\alpha$  para el cual la ecuación sea estable y estudia la evolución de la ecuación

$$x_{n+2} + (\alpha - 1)x_{n+1} + 0,25(\alpha^2 - 2\alpha + 1)x_n = n(1 - \theta(n - 10)) ,$$

donde  $\theta(k)$  es la función paso de Heaviside discreta, definida como

$$\theta(k) = \begin{cases} 1 & \text{si } k \geq 0 \\ 0 & \text{si } k < 0 \end{cases}$$

y suponemos como condiciones iniciales  $x_0 = 0$ ,  $x_1 = 0$ .

4. La ecuación de la transmisión del calor 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,  $x_1, x_2, \dots, x_{19}$ .

## Práctica 8

### Cálculo simbólico, derivación e integración con MatLab. Aplicación a la variable compleja.

El objetivo de esta práctica es utilizar el paquete MatLab para el cálculo integral, tanto en variable real como en variable compleja. Para ello será útil hacer uso del cálculo simbólico. Éste nos permitirá resolver ecuaciones y sistemas de ecuaciones, manipular expresiones algebraicas, calcular derivadas e integrales, trabajar con series, etc...

#### 8.1. Cálculo simbólico.

El cálculo simbólico es una forma de calcular que tiene el MatLab que permite la introducción de funciones, expresiones algebraicas y su posterior manipulación para simplificarlas, derivarlas, integrarlas, etc.

Para trabajar de manera simbólica, el MatLab necesita que previamente definamos todas las variables como simbólicas, para ellos utilizaremos el comando `sym('expresión')`, como veremos a continuación.

Antes de empezar a trabajar de manera simbólica detengámonos a comprobar que ocurre cuando tratamos de escribir una función en el MatLab directamente sin cálculo simbólico. Si introducimos

```
f=x^2
```

obtenemos el siguiente mensaje de error:

```
??? Undefined function or variable 'x'
```

Hagamos ahora lo mismo pero definiendo previamente la variable `x` como simbólica:

```
x=sym('x');  
f=x^2
```

en este caso el MatLab acepta la definición de `f` como función de `x` y podremos, por lo tanto, utilizarla como tal como veremos más adelante.

También podemos definir una o varias variables como simbólicas con el comando `syms` de la siguiente manera:

```
x=syms x;
```

o, en el caso de varias variables, separando éstas mediante espacios:

```
syms x y z;
```

Con esto podemos hacer diferentes manipulaciones algebraicas de expresiones polinómicas, trigonométricas, logarítmicas, etc..., como simplificar expresiones, factorizar, expandir, utilizando comandos como `simplify`, `factor` y `expand`. Veamos varios ejemplos:

Introduciendo el comando:

```
simplify((sin(x))^2+(cos(x))^2)
```

obtenemos la simplificación:

```
1
```

Podemos factorizar un polinomio de la siguiente manera:

```
factor(x^3+4*x^2-3*x-18)
```

el cual nos da:

```
(x-2)*(x+3)^2
```

De la misma manera podemos desarrollar una expresión algebraica utilizando el comando:

```
expand((x-2)^2*(x+1))
```

el cual desarrolla el producto obteniendo:

```
x^3-3*x^2+4
```

Como hemos visto anteriormente, el definir una variable como simbólica nos permite introducir diferentes funciones dependientes de esta variable de manera que si queremos evaluarlas bastaría utilizar el comando `subs`, el cual permite sustituir la variable simbólica por un valor numérico. Por ejemplo, la expresión:

```
syms x;  
f=x^2+3*x+log(x);  
subs(f,5,x);
```

nos da el valor numérico:

```
41.6094
```

También podemos sustituir expresiones algebraicas en las funciones. Por ejemplo si consideramos la función compleja  $f(z) = z^3 + 2z + i$  y queremos descomponerla en su parte real e imaginaria, podemos introducir los siguientes comandos:

```
syms x y z;  
f=z^2+i;  
simplify(subs(f,x+i*y,z))
```

obteniendo

```
x^2-y^2+2*y*x+i
```

Esta última expresión se puede ver más comodamente en forma matemática con el comando `pretty`:

```
pretty(x^2-y^2+2*y*x*i+i)
```

el cual da

$$x^2 - y^2 - 2xyi + i .$$

Combiene recordar para el cálculo en variable compleja utilizando MatLab que todas las funciones elementales que tiene implementadas permiten trabajar en variable compleja, por el ejemplo, las funciones trigonométricas:

```
sin(2+3*i)
```

dará directamente como resultado:

```
9.1545 - 4.1689i
```

Análogamente para las funciones hiperbólicas, exponencial, etc.

En el caso de funciones multivaluadas para los números complejos, como puede ser el logaritmo, el MatLab tan sólo calcula su parte principal. Así por ejemplo, la expresión:

```
log(1/2 + sqrt(3)*i/2)
```

devuelve el resultado:

```
1.04721i
```

Por último, volviendo al cálculo simbólico, comentar que hay comandos en el MatLab que estan definidos tanto para cálculos numéricos como para simbólicos, con definiciones diferentes según el caso. Por ejemplo el comando `diff` calcula la derivada de una función en el cálculo simbólico, como podemos ver haciendo

```
help sym/diff
```

mientras que

```
help diff
```

nos dará su utilidad en el cálculo numérico.

## 8.2. Resolución de ecuaciones

Antes de pasar a ver cómo derivar e integrar funciones de forma simbólica utilizando MatLab, veamos cómo calcular la solución de una ecuación o sistema de ecuaciones. Para ello, utilizaremos el comando `solve` como vemos en los siguientes ejemplos.

Introduciendo el comando

```
solve('x^2-2*x-3')
```

obtenemos las raíces:

```
[-1]
[ 3]
```

También podemos utilizar aquí el comando `vpa` para obtener las raíces de una ecuación controlando el número de decimales que aparecen:

```
vpa(solve('x^3+2*x+24=0'),10)
```

dará como raíces:

```
-2.653912571
1.326956286 - 2.698599443i
1.326956286 + 2.698599443i
```

Trabajando en variable compleja, el comando `solve` nos servirá para encontrar todas las raíces  $n$ -ésimas de un número complejo  $z_0$ , ya que éstas se pueden obtener resolviendo una ecuación de la forma:

$$z^n - z_0 = 0 .$$

Así si, por ejemplo, queremos calcular las raíces quintas del número imaginario  $i$ , al introducir

```
i^(1/5)
```

obtenemos el resultado

```
0.9511+0.3090i
```

que corresponde únicamente a su parte principal. Sin embargo, si queremos obtener las cinco raíces quintas del número  $i$ , tendremos que resolver la ecuación  $z^5 - i = 0$ . Por tanto introduciremos el comando:

```
numeric(solve('z^5-i=0'))
```

para obtener las cinco raíces correspondientes:

```
-0.9511+0.3090i
-0.5878-0.8090i
 0.5878-0.8090i
 0.9511+0.3090i
 0+1.0000i
```

Por último, si queremos resolver un sistema de ecuaciones, la manera de introducirlo sería:

```
[x,y,z]=solve('x+y-3*i*z=0,x-3*y+2*z=1,x-y-z=2','x,y,z')
```

cuya solución resulta:

```
x = 13/10 - 9/10*i
y = -11/50 - 27/50*i
z = -12/25 - 9/25*i
```

y obteniendo la representación real de estas fracciones con el comando `numeric` tenemos:

```
numeric(x)
```

resulta

```
1.3000 - 0.9000i
```

y analogamente para las otras dos variables.

### 8.3. Derivación simbólica con MatLab

Como hemos visto anteriormente, el comando `diff` nos permite calcular derivadas de funciones con el MatLab. Para ello debemos escribir `diff(función, variable, n)`, donde la  $n$  indica el orden de la derivada que queremos realizar. Por defecto el MatLab toma  $n = 1$  si no introducimos dicho número.

Así por ejemplo, para funciones de una única variable, podemos introducir:

```
syms x;
f=x^3+2*x;
diff(f)
```

y obtenemos

```
3*x^2+2;
```

que es la primera derivada respecto de  $x$  de  $f(x) = x^3 + 2x$ . Si escribimos:

```
diff(f,x,3)
```

obtenemos 6, que la tercera derivada respecto de  $x$  de  $f(x)$ .

Para funciones de varias variables, las derivadas parciales se pueden calcular con MatLab del mismo modo. Por ejemplo, podemos escribir expresiones como:

```
syms x y;
f=(x^2+y^2)/(x-y);
g=diff(f,x,2)
h=diff(f,y,2)
k=diff(diff(f,x),y)
```

para obtener las derivadas parciales segundas de  $f(x)$  correspondientes.

Notad que el comando `syms x` o `syms x y` de los ejemplos anteriores podría ser suprimido si en su lugar utilizamos el comando `diff` como `diff('funcion','variable',n)`. Con los argumentos `funcion` y `variable` entre comillas, ésto evita el definir previamente las variables como simbólicas. Por ejemplo podemos escribir:

```
clear
diff('cos(4*x)','x',3)
```

para que calcule la derivada tercera de la función  $f(x) = \cos(4x)$ .

El cálculo de derivadas parciales resulta útil en el caso de funciones de variable compleja ya que, descomponiendo la función compleja en las funciones de dos variables parte real y parte imaginaria, podemos comprobar si se verifican o no las condiciones de Cauchy–Riemann.

## 8.4. Integración simbólica con MatLab

El MatLab permite calcular integrales analíticas y numéricas. De hecho puede calcular primitivas de funciones, así como realizar integrales definidas e impropias.

El comando a utilizar es el comando `int(funcion,variable,a,b)`, siendo `a` y `b` los extremos de integración en el caso de integrales definidas. Si la función es de una única variable podremos omitir ésta en el comando y si prescindimos de los límites de integración, calcula una primitiva del integrando.

Como en el caso de las derivadas, podremos omitir la definición de las variables como simbólicas si introducimos los argumentos `funcion` y `variable` entre comillas.

Por ejemplo, la instrucción:

```
int('cos(4*x)')
```

nos da el resultado

```
1/4*sin(4*x)
```

La integral definida  $\int_0^{\pi/8} \cos(4x)dx$  se calcularía del siguiente modo:

```
int('cos(4*x)',0, pi/8)
```

obteniendo el resultado:

```
1/4
```

Existen integrales que el MatLab no puede calcular, por ejemplo:

```
int('sin(x)/x')
```

no obstante, podemos evaluar integrales definidas de estas funciones utilizando el comando `numeric`, el cual permite transformar una variable simbólica en numérica. Es decir, en este caso el comando `numeric` permite evaluar numéricamente la integral. Por ejemplo, el comando:

```
numeric(int('sin(x)/x',2,3))
```

da como resultado:

```
0.2432
```

Las integrales dobles y triples se calculan utilizando el comando `int` sucesivas veces teniendo en cuenta que debemos introducir correctamente los límites de integración, los cuales dependerán de la región de integración. Para obtenerlos podemos ayudarnos del MatLab dibujando la región correspondiente con los comandos vistos en la práctica anterior.

Como nuestro objetivo es el cálculo de integrales de funciones de variable compleja, nos centraremos en cómo calcular este tipo de integrales curvilíneas.

Recordaremos en primer lugar que la integral curvilínea de una función  $f(z)$  continua en un dominio  $D \in \mathbb{C}$  a lo largo de una curva suave a trozos  $z(t)$ , cuya gráfica  $C$  está contenida en  $D$ , viene dada por

$$\int_C f(z)dz = \int_a^b f(z(t))z'(t)dt.$$

Así, si queremos calcular la integral

$$\int_C z^2 dz$$

siendo  $C$  el arco de circunferencia de centro  $(0,0)$  y radio 2 situado en el primer cuadrante y recorrido en sentido positivo, procederemos de la siguiente manera:

Primero parametrizaremos la curva  $C$  de la forma

$$\begin{aligned}x(t) &= 2 \cos(t) \\ y(t) &= 2 \sin(t)\end{aligned}$$

la cual podemos dibujar con el MatLab utilizando el comando `plot` visto en la práctica anterior. Su representación nos puede ayudar a la determinación del intervalo de variación

de la variable  $t$ , aunque en general no será necesario representarla. En nuestro caso, el intervalo de variación de la  $t$  será  $[0, \pi/2]$ .

Por lo tanto la integral se calculará introduciendo en el Matlab los siguientes comandos:

```
syms t z;
x=2*cos(t);
y=2*sin(t);
f=z^2;
g=simplify(subs(f,x+i*y,z));
int(g*(diff(x)+i*diff(y)),0,pi/2)
```

obteniendo que el resultado de dicha integral curvilínea es

$$-\frac{8}{3} - \frac{8}{3}i.$$

## 8.5. Ejercicios

1. Construye una función que calcule todas las raíces de orden  $n$  de un número complejo  $z$  y te las de en forma de vector. Utilízala para calcular todas las raíces décimas de 1.
2. Resuelve la ecuación  $(z^3 - 27)(z^5 + 1) = 0$ .
3. Obtén la descomposición de la función  $f(z) = iz^3 + 2z$  en su parte real y su parte imaginaria y comprueba si cumple las condiciones de Cauchy-Riemann. Es holomorfa?
4. Comprueba si las siguiente funciones son holomorfas y en caso afirmativo calcular su derivada:
  - a)  $f(z) = z$
  - b)  $f(z) = \sinh\left(\frac{z}{i}\right)$
  - c)  $f(z) = \sin(z^2)$
5. Dibuja los puntos  $z = i$ ,  $z = -1 + i$ ,  $z = -2i$ ,  $z = 5$  y el circuito formado por la poligonal que une dichos puntos. A continuación calcula, parametrizando la curva, la integral

$$\int_C \frac{dz}{z}$$

79

siendo  $C$  el circuito anterior recorrido en sentido positivo.

6. Dada la circunferencia  $C$  de centro  $z = 3$  y radio 1, calcula el valor de la integral

$$\int_C \frac{1}{(z-3)^2} dz$$

considerando  $C$  recorrida en sentido positivo.

80