1 Una introducción a MATLAB

PRÁCTICA NÚMERO 1

Introducción al MATLAB

Matrices y vectores

Los llamados vectores permiten almacenar colecciones de datos. Por ejemplo

```
»a=[3 2 1]
```

define un **vector fila** denominado a que consta de 3 componentes. En cada una de las componentes se almacena un número. Si hubieramos deseado introducir un **vector columna** cada uno de los números tendría que estar separado por ;

```
»b=[3; 2; 1]
```

El producto de matrices se denota por *

```
»C=a*b
```

y se comprueba que no es conmutativo

```
»d=b*a
```

En cualquier momento podemos emplear los vecores ya definidos. Por ejemplo

```
»sqrt(a)
```

calcula la raíz cuadrada de los elementos del vector a.

Si queremos hacer operaciones aritmética `+,-, * ,/` componente a componente debemos preceder el símbolo por un punto .

```
»a./2
```

dividirá las componentes del vector a por 2.

Si se quiere referir a una componente de un vector determinado se indica con `nombrevector(posicion)`. Por ejemplo si queremos cambiar el valor de la primera componente del vector a al valor 10, escribiremos:

```
»a(1)=10
```

Una forma de crear rápidamente un vector es con la instrucción `a:incremento:b`. Genera un vector con los números comprendidos en $[a, b]$, iniciándose en `a` y espaciados la cantidad `incremento`. Por ejemplo

```
»x=0:0.1:1
```

Genera el vector $x = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$

Una **matriz** es un vector de vectores. Por ejemplo si escribimos

```
»B=[1 2 3; 4 5 6; 7 8 9]
```

Introducimos una matriz llamada B. A esta matriz (vector de vectores) le podemos añadir nuevos vectores filas o columna. Por ejemplo si quisiéramos añadir el vector columna b a la matriz B lo haríamos con la sentencia:

```
»D=[B ,b]
```

Si queremos añadir una nueva fila a la matriz B escribiríamos la sentencia:

```
»E=[B;a]
```

Notar que en las dos expresiones anteriores podíamos usar el nombre del vector o su definición explícita. Notar además que b es un vector columna y que a es un vector fila.

Visualización 2D

En esta sección analizaremos los gráficos en 2 dimensiones. Los gráficos en dos dimensiones más básicos consisten en unir consecutivamente un conjunto de puntos (x_i, y_i). Consideremos el siguiente ejemplo

```
»x=[1.5 2.2 3.3 4.4 5.8 6.7 7.3 8.3];
»y=[2.5 3.4 4.3 4.4 6.8 3.7 2.3 1.3];
```

la sentencia siguiente dibuja consecutivamente los anteriores puntos `(x(i),y(i))`

```
»plot (x,y)
```

Se puede especificar un formato a las gráficas mediante la sentencia:

```
»plot(x,y,'caracteres')
```

`'caracteres'` es un *string* que contiene las especificaciones del color, del tipo de marca y del tipo de línea del dibujo. Por ejemplo

`»plot(x,y,'r+-')` dibujaría la gráfica en rojo, empleando como marcas el símbolo `+` y haciendo un trazo discontinuo. Estos argumentos se puede escribir en cualquier orden.

Además si suprimimos la indicación del tipo de línea `plot(x,y,'r+')` no dibujaría la línea entre los puntos.

También se pude poner varias gráficas a la vez. Por ejemplo:

```
»plot(x,y,'r+-',a,b,'r-')
```

dibujaría dos gráficas, la asociada a (x_i, y_i) y a (a_i, b_i).

El comando `plot` también acepta como entrada una matriz en lugar de un vector. Si denotamos `X` e `Y` dos matrices, las siguient expresión `plot(x,Y)` dibujaría simultáneamente todas las columnas de `Y` respecto a la variable `x`. Si hubieramos escrito `plot(X,Y)` dibujaría todas las columnas de `X` con todas las columnas de `Y`. Si solamente recibe un argumento `plot(Y)` dibujaría todos los vectores respecto a los índices, esto es, representaría una sucesión.

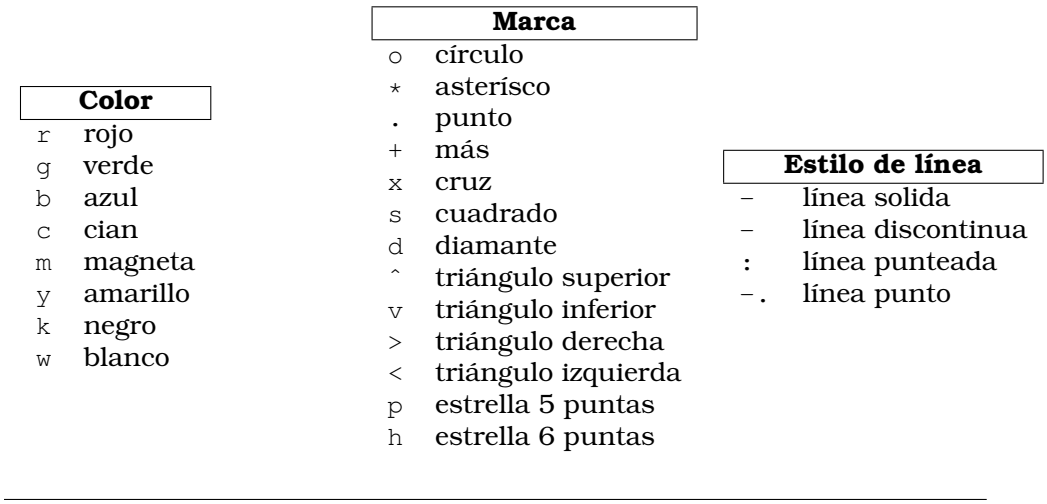

Figura 1.1: Opciones del comando `plot`

Los mandatos `LineWidth` permite cambiar el grosor de la líneas, con `MarkerSize` el tamaño, `MarkerEdgeColor` el contorno de las marcas y `MarkerFaceColor` el relleno de la marca.

```
plot(x,y,'LineWidth',3,'MarkerSize',5, 'MarkerFaceColor','g')
```

La tabla 1.2 recoge ciertas funciones para gestionar los ejes y anotaciones de las figuras. Las tabla 1.3 muestra algunos comandos para gestionar las ventanas de dibujo.

Hasta aquí se ha visto el mandato `plot` pero existen muchas otras funciones para realizar gráficos en 2 dimensiones. La tabla 1.4 recoge alguna de estas funciones:

Se puede hacer **varios dibujos** en la misma figura. Cada uno de estos dibujos puede realizarse con una función de la tabla 1.4. Esta posibilidad es mediante el comando

```
»subplot(m,n,p)
```

se particiona la figura de dibujo en una matriz $m \times n$ y mediante el parámetro p se elige una de las $m * n$ celdillas. Por ejemplo `subplot(4,2,7)` se refiera a la gráfica de la posición $(4,1)$. Si escribimos:

```
»subplot(2,1,1), fplot('sin(x)',[0, 2*pi],'-')
»subplot(2,1,2), fplot('cos(x)',[0, 2*pi],'-')
```

Alternativavemente a la función `fplot` se puede dibujar funciones elementales a partir de la construcción de una tabla de datos. Por ejemplo si quisieramos dibujar la función $y = \cos(x)$ en el intervalor $[-\pi, \pi]$ escribiríamos:

```
»x=-pi:0.1:pi, ycos=cos(x), plot(x,ycos)
```

Las instrucciones vienen separadas por una coma. La primera instrucción crea un vector denominado `x` que cuya primera componente es $-\pi$ y el resto de coordenadas van aumentando su valor en 0.1 unidades hasta sobrepasar el valor de π. La segunda instrucción crea el vector `ycos` que contiene los

`axis ([xmin xmax y min ymax])`	límites de los ejes x e y
`axis auto`	límites de los ejes por defecto
`axis equal`	estandariza las unidades de los ejes x-, y- z-
`axis on/off`	inserta/elimina los ejes
`grid on/off`	inserta/elimina una rejilla en la figura
`axis square`	hace una caja cuadrada de ejes
`axis tight`	fija los ejes relativos a un conjunto de datos
`xlim([xmin xmax])`	fija los límites del eje x-
`ylim([ymin ymax])`	fija los límites del eje y-
`box off`	elimina el cuadrado que enmarca la figura
`legend('expr1',..,'exprn',n)`	leyenda de la figura. $n = \{-1, 0, 1, 2, 3, 4\}$ cambia posición?
`xlabel('texto x',opciones)`	escribe el titulo eje x- con opciones
`ylabel('texto y',opciones)`	escribe el titulo eje y- con opciones
`title('texto figura',opciones)`	escribe el titulo de la figura con opciones
`text(x,y,'texto',opciones)`	inserta en la posición (x,y) el texto con opciones

Figura 1.2: Ejes y anotaciones

`hold on`	incluye todos los dibujos en una figura
`hold off`	deriva cada comando `plot` a figuras diferentes
`clf`	limpia la figura actual
`close`	cierra la figura
`close all`	cierra todas las figuras
`figure(n)`	crea la figura n

Figura 1.3: Algunos comandos para gestionar las ventanas de figuras 2D

valores de la función coseno evaluada componente a componente en el vector `x`. Finalmente dibujamos la taba de datos `x,ycos`.

Help

Una forma rápida de consultar las posibilidades y sintaxis de una determinada función de MATLAB es utilizar la función help `help NombreFuncion` (ejemplo: help plot). También se puede utilizar la interfaz gráfica para buscar ayuda entrando en Help-Product Help.

M-Ficheros

La forma natural de trabajar con Matlab es a través de los llamados **M-ficheros**. Estos ficheros contienen un conjunto de instrucciones Matlab que se pueden invocar su ejecución desde la línea de comandos. Trabajar con este tipo de ficheros ahorra tiempo en el desarrollo del programa definitivo,

`plot`	Dibujo básico 2D
`loglog`	Dibujo 2D con escalas logarítmicas
`semilogx`	Dibujo 2D con escala logarítmica eje x-
`semilogy`	Dibujo 2D con escala logarítmica eje y-
`semilogx`	Dibujo 2D con escala logarítmica eje x-
`plotyy`	Dibujo 2D con ejes y- a la izquierda y derecha
`polar`	Dibujo 2D en coordenadas polares
`fplot`	Dibuja automáticamente una función
`ezplot`	Versión sencilla de `fplot`
`ezpolar`	Versión sencilla de `polar`
`fill`	Rellena un polígono
`area`	Rellena el área de una gráfica
`bar`	Gráfico de barras
`barh`	Gráfico de barras horizontal
`hist`	Histograma
`pie`	Gráfico de tarta
`scatter`	Gráfico de puntos

Figura 1.4: Funciones para hacer gráficos 2D

permitiendo introducir modificaciones y ejecutarlas instántaneamente. Además permiten grabar exactamente lo que se hizo y poderlo reproducir en otra ocasión, por uno mismo o por otros compañeros. Además se pueden crear utilidades que son reutilizables. Este tipo de ficheros son de dos tipos:

- **Scripts.** Son un conjunto de instrucciones que no necesitan unos datos de entrada y operan con las variables allí contenidas.

- **Funciones.** En estos ficheros se introducen/devuelven datos. El resto de datos son de uso local (interno a la función) y no son accesibles desde el resto de los M-ficheros.

Los M-ficheros se pueden crear con cualquier editor de texto. Es recomendable el propio editor de Matlab. Una vez creado para poderse usar debe estar en una lista de directorios de búsqueda. Con el comando `path` se puede ver el actual directorio de búsqueda. Desde la ventana se puede elegir un directorio de búsqueda.

A continuación listamos un pequeño ejemplo de M-fichero (*script*) para dibujar la función $y = \text{sen}(x)$ e $y = \cos(x)$ en el intervalo $[-\pi, \pi]$ y dibujadas en dos gráficas separadas.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   E j e m p l o   d e    M f i l e
%   Este fichero es un script que dibuja la funcion
%   seno y coseno en el intervalo [-pi,pi]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% El signo % (porcentaje) permite introducir comentarios
% Generamos valores de la variable independiente x
% en el intervalo [-pi,pi] espaciados 0.1 unidades
x=-pi:0.01:pi;
% calculamos las imagenes del seno y coseno y lo almacenamos en el
% vector yseno e ycoseno
yseno=sin(x);
ycoseno=cos(x);
% Dibujamos dos figuras.  Una contiene la gráfica de la función
% seno y la otra de la función coseno. Las gráficas se dibujan en
% negro, con un grosor de 2 unidades y los puntos unidos.
subplot(1,2,1),plot(x,yseno,'k-','LineWidth',2)
subplot(1,2,2),plot(x,ycoseno,'k-','LineWidth',2)
```

Figura 1.5: Ejemplo de M file: **EjemploMfile.m**

-

GUIÓN DE LA PRÁCTICA

1. Dibuja un cuadrado de vértices $(1,1),(1,-1),(-1,-1),(-1,1)$
2. Se considera la siguiente matriz:

$$R(\alpha) := \begin{pmatrix} \cos\alpha & \text{sen}\alpha \\ -\text{sen}\alpha & \cos\alpha \end{pmatrix}$$

Esta matriz permite rotar el punto $(x_2, y_2)^T$ un ángulo α en el sentido de las agujas del reló como ilustra la siguiente figura: Construye una

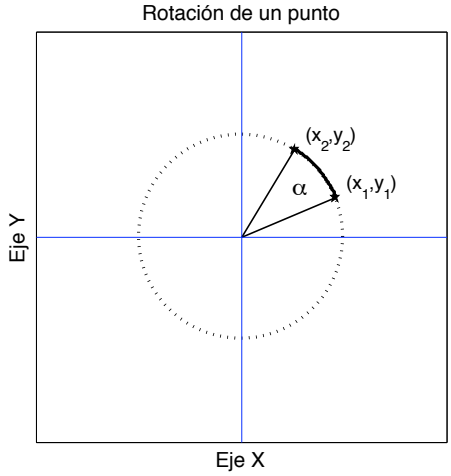

Figura 1.6: Resultado del script **RotacionPunto.m**

función que dado el punto $x = (x_2, y_2)^T$ y el ángulo α te calcule el punto rotado (x_1, y_1) mediante la expresión:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\alpha & \text{sen}\alpha \\ -\text{sen}\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

3. Construye un M-fichero de Matlab que dibuje un cuadrado rotado un ángulo α en el sentido de las agujas del reló.
4. Construye un M-fichero de Matlab que dibuje un polígono regular de n lados.

EJERCICIOS PROPUESTOS

1. Define una matriz 5×5 de doses.
2. Introduce los vectores $(1, 2, \cdots, 100)$, $(\sin(1), \sin(2), \cdots, \sin(100))$, $(0, 0.1, 0.2, , \cdots, 10)$.
3. Dado un valor de n, crea un M-fichero para definir la siguiente matriz:

$$A = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ n+1 & n+2 & n+3 & \cdots & 2n \\ & & \cdots & & \\ n(n-1)+1 & n(n-1)+2 & n(n-1)+3 & \cdots & n^2 \end{pmatrix}$$

4. A partir de la matriz anterior y para $n = 5$ intercambia la fila 1 con la fila 4. Haz lo mismo con las columnas 3 y 5.
5. Calcula el valor de $A + A^2 + A^3$.
6. Calcula las gráficas de las funciones $y = \cos(x), y = \sin(x)$ e $y = \tan(x)$ sobre una misma figura en el intervalor $[0, 2\pi]$.
7. Emplea el comando `subplot` para repetir el ejercicio anterior pero en una única figura con las tres gráficas separadas.

PRÁCTICA NÚMERO 2

Programación I

Operadores relacionales y lógicos

En la realización de programas se requiere de **operadores lógicos y relacionales** que evalúan si una cierta expresión es verdadera o falsa. En función del resultado obtenido se realizará un conjunto de sentencias diferentes. Los operadores relacionales empleados por Matlab son:

Operadores relacionales	
==	igual
~ =	no igual
>	mayor que
<	menor que
>=	mayor o igual

Operadores lógicos	
&	y lógico
|	o lógico
~	negación lógica que
xor	o exclusivo lógico
all	verdadero si todos los elementos son no nulos
any	verdadero si alguno de los elementos son no nulos

Figura 1.7: Operadores relacionales y lógicos

Veamos algunos ejemplos si quisieramos testar si un número b cumple la relación $a \leq b \leq a^2$ se pueden testar del siguiente modo:

```
»a=2;b=3; (a <= b) & (b<= a*a)
```

la respuesta sería `ans=1` que indica que es verdadera para dicho par de números.

Estos operadores también trabajan sobre vectores y matrices, evaluando componente a componente. Considérese el siguiente ejemplo construido utilizando la función `ones(n)` que genera una matriz de orden $n \times n$ llena de unos:

```
»A=[1 2 ; 3 4]; B=2*ones(2)
»A==B
```

Respondería una matriz con tres 0 y un 1 que indicaría que solo la componente $(1,2)$ coincide.

Bucles

Los bucles `for` son instrucciones que permiten ejecutar un bloque de instrucciones para diferentes valores de ciertas variables. La estructura de estas sentencias son:

```
for   "variable"="expresión"
      instruccion 1
      ...
      instruccion n
end
```

La forma habitual para "expresión" es del tipo `i:s:j` que significa que "variable" iniciará su valor en `i` hasta alcanzar el valor `j`, aumentando su valor de `s` en `s` unidades. Por ejemplo si queremos sumar los números impares comprendidos en el intervalo $[1, 100]$ escribiríamos:

```
»suma=0; for i=1:2:100, suma=suma+i;end, suma
```

Notar que cuando la instrucción se introduce en la líneas de comandos en lugar de situarse en un m-fichero se separa por comas.

Si se omite el argumento `s` los incrementos serán de uno en uno. Por ejemplo si quisieramos sumar los cien primeros números naturales escibiríamos:

```
»suma=0; for i=1:100, suma=suma+i;end, suma
```

Otra forma de definir la "expresión" es mediante los vectores []. Por ejemplo si queremos obtener una tabla de valores de la función $f(x) = x^2$ para los datos contenidos en el vector [0 0.5 1] escribiríamos:

```
»for x=[0 0.5 1], disp([x x*x]), end
```

Se pueden anidar múltiples bucles. Por ejemplo si queremos construir una matriz 5×5 que en la posición (i, j) contenga el valor $(-1)^{i+j}$ escribiríamos:

```
n=5; A=eye(n);
for i=1:n
 for  j=1:n
      A(i,j)=(-1) ^(i+j)
 end
end
```

Si el número de veces que se debe ejecutar un conjunto de sentencias no es conocido a priori se puede recurrir al bucle `while`. Este tiene la forma:

```
while   "condición"
        instruccion 1
        ...
        instruccion n
end
```

Este conjunto de sentencias se ejecutará mientras "condición" sea verdadera. Por ejemplo si quisiéramos escribir los números cuadrados perfectos menor que 1000 escribiríamos:

```
n=1;
while (n^2<1000)
 disp(n^2),
 n=n+1;
end
```

Control de flujo

En la ejecución de programas ciertas instrucciones deben ejecutarse dependiendo de ciertas circunstancias. Esto se puede controlar mediante la instrucción `if`. Esta tiene la siguiente estructura:

```
if    condición
      instruccion 1
      ...
      instruccion n
end
```

También se puede particionar la ejecución de un programa en bloques de instrucciones que se ejecutan según cierto conjunto de condiciones. La estructura es:

```
if (condición 1)
   Bloque 1 de instrucciones
elseif (condición 2)
   Bloque 2 de instrucciones
   ...
else
   Bloque n (último) de instrucciones
end
```

El siguiente *script* implementa un código para dibujar la gráfica de la función valor absoluto en el intervalo $[-1, 1]$:

$$|x| := \begin{cases} -x & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases}$$

Programación en MATLAB

MATLAB es un lenguaje interpretado. Esto significa que para ejecutar una instrucción ésta debe ser leída y descodificada antes de efectuar su acción. Para ahorrar tiempo de ejecución es fundamental trabajar con expresiones vectoriales. Esta forma de programar la vamos a ilustrar con un ejemplo. Vamos a dar una programación alternativa del M-fichero que dibuja el valor absoluto.

Una función importante para trabajar con vectores es la instrucción `find` que devuelve las coordenadas (o índices) de un vector cuyos valores son no ceros. Por ejemplo,

```
»x=[1,0,2,0,4,-inf];f=find(x)
```

devuelve el valor `f=[1,3,5,6]` que son las cuatro componentes diferentes de cero. También se puede utilizar par extraer estas componentes con la instrucción `x(f)`. También se puede aplicar esta instrucción a matrices. A partir de una determinada matriz se forma un vector consistente en concatenar todos los vectores columnas y trabaja sobre el vector resultante:

```
»A=[1 1 1;2 2 2];B=[2 2 2;1 1 1]; f=find(A<B)
```

la respuesta obtenida es `f=[1 3 5]` y se puede emplear para cambiar el valor de estas coordenadas por ejemplo

```
»A(f)=0
```

que pondrá su primera fila a cero.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   E j e m p l o   V A L O R   A B S O L U T O
%   Este script gráfica del valor absoluto en [-1,1]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x=-1:0.1:1;
y=[];
n=length(x);
for i=1:21
if (x(i) <0)
y(i)=-x(i)
else
y(i)=x(i)
end
end
plot(x,y,'-k','LineWidth',2);
```

Figura 1.8: Script para dibujar la gráfica del valor absoluto **valorabsoluto1.m**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   E j e m p l o   V A L O R   A B S O L U T O
%   Script alternativo para dibujar la gráfica del
%   valor absoluto en [-1,1]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x=-1:0.1:1
y=x;
y(find(x<0))=-x(find(x<0));
plot(x,y);
```

Figura 1.9: Script alternativo para dibujara la gráfica del valor absoluto **valorabsoluto.m**

Por ejemplo para dibujar la función valor absoluto con un programación vectorial podemos hacer el siguiente M-Fichero:

GUIÓN DE LA PRÁCTICA

1. **Modelo A.** Considérese el siguiente modelo matemático para describir la evolución de la población de ballenas

$$\boxed{P(t+\Delta t) = P(t) + [(N - M_n)P(t) - M_p(P(t))]\,\Delta t}$$

donde

t: es la variable tiempo t.

Δt: Intervalo de tiempo entre dos instantes en los que se analiza el sistema.

N: Tasa anual de nacimientos por ballena. Si multiplicamos por 1000 esta tasa, el valor $1000N$ representaría el número de nacimientos habidos en un año por cada 1000 ballenas existentes.

Mn: Tasa anual de muertes por ballena. Es equivalente a la tasa de natalidad pero para la mortalidad de las ballenas.

Mp: Ballenas pescada anualmente. Es una decisión que depende del número ejemplares existentes.

$P(t)$: Número de ballenas en el instante t.

Esta ecuación dice que el número de ballenas que hay en el instante $t + \Delta t$ son las que había en el instante t más las que han nacido menos las que han muerto por muerte natural o por pesca.

Esta ecuación nos permita obtener puntos de gráfica de la función $P(t)$ (evolución de la población de ballenas en el tiempo). Estos puntos se guardaran en un vector para posteriormente dibujarlos.

$$\begin{aligned} t_i &= t_0 + i\Delta t \\ P_i &= P(t_i) \\ Mp_i &= Mp(P_i) \end{aligned}$$

Modelo A $\boxed{P_{i+1} = P_i + [(N - Mn)P_i - Mp_i]\,\Delta t}$

La ecuación anterior es un ejemplo de sucesión recursiva en el que el valor del término $(i+1)-$ésimo se calcula en función del témino $i-$ésimo.

Se quiere aplicar el anterior modelo para estudiar las siguientes políticas de pesca:

- POLÍTICA I: $Mp(x) := 0$ (se prohibe la pesca de ballenas)
- POLÍTICA II: $Mp(x) := 3000$ (sistemáticamente se permite la pesca de 3000 ejemplares)
- POLÍTICA III: $Mp(x) := 0.1x$ (se captura anualmente el $10\,\%$ de los ejemplares existentes).

- POLÍTICA IV: $Mp(x) := min(2000, 0.1x)$ (se permite a la industria pesquera la captura anual del $10\,\%$ de los ejemplares garantizándole una captura mínima de 2000 ejemplares)

Considerese el siguiente conjunto de parámetros:

$$\begin{aligned} \Delta t &= 0.25 \text{ año} \\ N &= 2485/22500 \\ M_n &= 1125/22500 \\ P_0 &= 25000 \end{aligned}$$

Ejercicio. Construye una función Matlab para calcular la evolución de la población de ballenas en los próximos 25 años. Representa simúltaneamente las cuatro políticas consideradas. ¿Qué comportamientos son pocos realistas? ¿Cual es la cantidad máxima de capturas que no conduce a la extinción de las ballenas?

2. **Modelo B.** Ahora introduciremos dos nuevos factores en el modelo.

   Supondremos que si la población de ballenas es demasiado elevada los recursos alimenticeos empiezan a escasear y la tasa normal de mortalidad empieza a incrementarse de acuerdo con el siguiente multiplicador:

$$MultM(P) := \begin{cases} 1 & \text{si } P \leq 30000 \\ 1 + 2\left(\frac{P-30000}{30000}\right) & \text{si } 30000 \leq P \leq 60000 \\ 3 + 4\left(\frac{P-60000}{60000}\right) & \text{si } P \geq 60000 \end{cases}$$

   Este multiplicador indica que a partir de 30000 ejemplares la tasa de mortalidad se empieza a incrementar (muerte por desnutrición) proporcionalmente llegando a una tasa tres veces más elevada en los 60000 ejemplares. Esta tasa proporcionalidad se duplica al sobrepasar los 60000 ejemplares.

   Por otro lado supondremos que si la población de ballenas es demasiado pequeña existen dificultades en encontrar parejas reproductoras entre las ballenas y este hecho influye en la tasa de natalidad de acuerdo al siguiente multiplicador:

$$MultN(P) := \begin{cases} \frac{P}{15000} & \text{si } P \leq 15000 \\ 1 & \text{si } P \geq 15000 \end{cases}$$

   Este multiplicador indica que a partir de los 15000 ejemplares el efecto de la escasez de ballena empieza a aparecer. Asumimos un efecto proporcional, llegando a no reproducirse cuando no existen ejemplares.

   El modelo resultante queda:

$$P(t + \Delta t) = P(t) + [(N * MultN(P(t)) - M_n * MultM(P(t)))P(t) - M_p(P(t))]\,\Delta t$$

   O expresado como una sucesión recursiva:

Modelo B	$P_{i+1} = P_i + [(N * MultN(P_i) - M_n * MultM(P_i))P_i - Mp_i]\,\Delta t$

Ejercicio. Introducir estas modificaciones en el modelo y simular nuevamente los anteriores escenarios. Hacer dos dibujos simultáneamente para cada uno de los modelos.

EJERCICIOS PROPUESTOS

1. Haz un M-fichero para sumar dos matrices de dimensión 2×3 definidas anteriormente.
2. Modifica el anterior fichero para que las matrices sean solicitadas al usuario. Ayuda funcion `x=input('Entre un número: ');`
3. Modifica el programa para que el usuario entre las matrices A y B de una sola vez y devuleva la matriz $C = A + B$.
4. Crea un M-fichero para calcular el elmento mínimo de una matriz, la columna y la fila donde se encuentra.
5. Modifica el M-fichero anterior para mostrar todos los elementos mínimos en caso de alcanzarse en varias celdillas de la matriz.
6. Dibuja la siguiente función definida a trozos:

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

PRÁCTICA NÚMERO 3

El concepto de función

Históricamente el concepto de función ha estado asociado a una fórmula explícita del tipo $y = f(x)$. Ejemplos de estas funciones son las funciones polinómicas como $f(x) = x^2 + x + 1$ o $f(x) = x^3 + 1$ o funciones trigonométricas del tipo $f(x) = \cos x$ o $f(x) = \text{sen}\, x$. La llegada de los ordenadores ha permitido trabajar con expresiones mucho más generales y trabajar con funciones sin fórmula, sino como procesos que permiten asignar a un objeto matemático a otro objeto, no necesariamente del mismo tipo. El programa Matlab recoge todas estas posibilidades.

Consideremos la siguiente fórmula que permite calcular la cuota de una hipoteca en función de varios datos:

$$H = C\frac{i(i+1)^n}{(i+1)^n - 1} \tag{1.1}$$

donde

C: dinero del préstamo

i: interés del préstamo de cada mensualidad

n: número de mensualidades

Por ejemplo, si se quiere solicitar un préstamo hipotecario de 100000 euros durante 15 años a una tasa de interés del $3\,\%$ para calcular la cuota que mensualmente deberíamos pagar al banco tendríamos que $n = 15 * 12 = 180$ mensualidades a un interés mensual de $i = \frac{3/100}{12}$, y por tanto la hipoteca a pagar cada mensualidad sería:

$$H = 100000\frac{\frac{3/100}{12}(\frac{3/100}{12}+1)^{180}}{(\frac{3/100}{12}+1)^{180}-1} = \tag{1.2}$$

Esta fórmula es un ejemplo de función "clásica " dependiente de varias variables $H(i, n, C)$ y pude ser definida en Matlab, empleando un M-fichero, para su utilización:

Una vez definida se puede hacer uso de ella. Si quisieramos calcular el valor del ejemplo anterior haríamos:

```
»ejemplo=@hipoteca
```

posteriormente hacemos una llamada a la función para los argumentos definidos

```
»ejemplo(100000,3/1200,180)
```

Una función puede asignar un vector a un escalar (como en el ejemplo anterior devuelve el valor H) o puede devolver otro vector o incluso una

```
function H= hipoteca(C,i,n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Ejemplo de  función:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) N o m b r e:
%       hipoteca
% (2) D a t o s    d e   e n t r a d a:
%       C capital prestado
%       i interes mensual
%       n numero de mensualidades
% (3) D a t o s    d e   s a l i d a:
%       H hipoteca en cada mensualidad
% (4) P r o c e d i m i e n t o   d e   c á l c u l o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

H=C*(i*(i+1)^n)/((i+1)^n-1);
```

Figura 1.10: Ejemplo de función **hipoteca.m**

colección de argumentos. Para ilustrar esto último considerese los siguiente dos problemas: i) calcular la tabla de amortización del préstamo y ii) calcular la tabla de amortización del préstamo haciendo una cancelación parcial de $Camor$ en el periodo k. Las siguientes figuras muestran las funciones creadas para cada objetivo.

```
function T= tablaAmor(C,i,n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Esta función calcula la tabla de amortización de una hipoteca
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) N o m b r e:
%       tablaAmor
% (2) D a t o s    d e   e n t r a d a:
%       C capital prestado
%       i interes mensual
%       n numero de mensualidades
% (3) D a t o s    d e   s a l i d a:
%       T es un vector que contiene en la componente j-esima
%         el dinero pendiente de amortizar
% (4) P r o c e d i m i e n t o   d e   c á l c u l o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Hip=hipoteca(C,i,n); % cálculo de la hipoteca empleando
                        % la función definida
T=[];
T(1)=C;          % en el instante inicial se debe todo el
                        % capital
for j=1:n,
    T(j+1)=T(j)-Hip+T(j)*i;
                    % capital pendiente en el periodo j+1 es
                    % capital en el periodo j menos la hipoteca
                    % más lo interéses a pagar en dicho periodo j
end
end
```

Figura 1.11: Ejemplo de función que devuelve un vector **tablaAmor.m**

```
function [T,Hnueva]= tablaAmorPar(C,i,n,k,Camor)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Esta función calcula la tabla de amortización de una hipoteca
%  teniendo en cuenta que se realiza una amortización anticipada
%  de Camor euros en la k-ésima mensualidad
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) N o m b r e:
%       tablaAmorPar
% (2) D a t o s    d e   e n t r a d a:
%       C capital prestado
%       i interes mensual
%       n numero de mensualidades
%       k periodo en el que se realiza la amortizacion parcial
%       Camor cantidad amortizada anticipadamente
% (3) D a t o s    d e   s a l i d a:
%       T tabla de amortización
%       Hnueva nueva hipoteca tras realizar la amortización parcial
% (4) P r o c e d i m i e n t o   d e   c á l c u l o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Hip=hipoteca(C,i,n); % Cálculo de la hipoteca
T(1)=C;              % En el instante inicial
                     % se debe reintegrar todo el capital
for j=1:n
if j== k
T(j+1)=T(j)-Hip+T(j)*i-Camor;
Hnueva=hipoteca(T(j+1),i,(n-k));
        Hip=Hnueva
else
T(j+1)=T(j)-Hip+T(j)*i;
end
    end
end
```

Figura 1.12: Ejemplo de función que devuelve dos argumentos **tablaAmorPar.m**

```
function y_prima=DeriApro(f,x,h)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DERIV_APR calcula aproximadamente la derivada de
% la función f en el punto x tomando como
% incremento h o en caso de que este parámetro no se especifique
% se toma el valor h=SQRT(EPS), esto es,  un valor casi 0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) N o m b r e:
%       DeriApro
% (2) D a t o s    d e   e n t r a d a:
%       f función a derivar
%       x punto donde se aproxima la derivada
%       h parámetro para el cálculo de la derivada
% (3) D a t o s    d e   s a l i d a:
%       y_prima valor aproximado de f'(x)
% (4) P r o c e d i m i e n t o   d e   c á l c u l o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin < 3       % si el número de argumentos es menor de 3
    h=sqrt(eps);    % se toma un valor de h por defecto
end
y_prima=(f(x+h)-f(x))/h;
end
```

Figura 1.13: Ejemplo de función donde un input es una función. **DeriApro.m**

Supongamos que necesitamos crear una función para calcular aproximadamente la derivada

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

La función Matlab requiere tres argumentos: i) el punto x donde calculamos la derivada, ii) el incremento h considerado y por último iii) la función f considerada. Como resultado devolvera aproximadamente el valor de la derivada. Pues bien, Matlab tiene la posibilidad de pasarle la función como un argumento más. Para ilustrar este procedimiento consideramos el siguiente ejemplo:

En este ejemplo el primer argumento f es una función. En los ejemplos previos los argumentos de entradas solamente eran datos (un número, o un vector, o una matriz). La primera observación es que aparece la función feval que permite evaluar una función en sus argumentos. En esta caso evalúa una función real en los puntos $x+h$ y en el x.

Para hacer uso de esta función tenemos dos formas:

```
»DeriApro(@mifuncion,0,00.1)
```

Esta función, mifuncion, está definida en un m.fichero y es la que empleará. Por ejemplo podríamos calcular aproximadamente la derivada de la función $f(x) = \cos x$ en el punto 0 mediante la expresión:

```
»DeriApro(@cos,0,00.1)
```

También se puede crear una función anónima $f(x) = \exp(\cos(x))$desde la línea de comandos mediante la expresión:

```
function y=FunGlobal(x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FunGlobal ilustra el uso de variables globales y locales
%           en la definición  de una función
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) N o m b r e:
%       FunGlobal
% (2) D a t o s    d e   e n t r a d a:
%       x variable independiente (número real)
% (3) D a t o s    d e   s a l i d a:
%       y variable idependiente (número real)
% (4) P r o c e d i m i e n t o   d e   c á l c u l o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global A                % variable global
c=1;                    % variable local, solo accesible dentro
                        % de la definición de la función
y=c/(x+A);
end
```

Figura 1.14: **FunGlobal.m**

```
»unafuncion=@(x) exp(cos(x))
```

y posteriormente emplearla

```
»DeriApro(unafuncion,0,00.1)
```

Alternativamente también se puede pasar con la función inline como muestra el siguiente ejemplo:

```
»unafuncion=inline('exp(cos(x))')
```

Se puede comprobar que se trata de una función evaluándola en el valor 2

```
»unafuncion(2)
```

En la definición de funciones se pueden emplear variables que serán de uso local. Esto es, que solo son modificables dentro de la ejecución de la propia función. Si se desea que ciertas variables sen accesibles desde otro procedimiento hay que emplear **variables globales**. Esto se consigue con la sentencia global tal como muestra el siguiente ejemplo:

Si queremos dibuar $f(x) = 1/(x+1)$ en $[0,1]$ escribimos:

```
»global A, A=1; fplot(@FunGlobal,[0 1])
```

Si cambiamos desde la linea de comandos el valor del parámetro $A = 2$ este tendrá un efecto en la definición de la función FunGobal ya que se trata de una variable global.

```
»A=2; fplot(@FunGlobal,[0 1])
```

Por contra, si lo que cambiamos es el valor de la variable local c fuera de la definición de la función, en la consola, este cambio no tendrá ningún efecto:

```
»c=2; fplot(@FunGlobal,[0 1])
```

GUIÓN DE LA PRÁCTICA

1. Método de la bisección. El método de la bisección es un procedimiento para calcular una raíz de la función $f(x)$, esto es, resolver la ecuación $f(x) = 0$. Este procedimiento asume que la función es continua en un intervalo $[a, b]$ y que la función cambia de signo en los extremos del intervalo ($f(a)f(b) < 0$). En cada iteración calcula el punto medio del intervalo actual y decide cual de los dos subintervalos sigue conteniendo una raíz eligiendo este subintervalo como el nuevo intervalo de la iteración siguiente. Cada vez que aplica una iteración la longitud del intervalo inicial se reduce a la mitad y tras n iteraciones tenemos que

$$\ell_n = \frac{\ell_0}{2^n}$$

donde ℓ_0 es la longitud del intervalo incial y ℓ_n es el intervalo tras realizar la $n-$ésima iteración. El siguiente pseudocodigo implementa este algoritmo:

Algorithm 1: Algoritmo de bisección

Data: Sea $f(x)$ una función continua en $[a, b]$ cumpliendo que $f(a)f(b) < 0$. Sea ε un parámetro de tolerancia empleado como criterio de paro.
Result: Una raíz m de la función $f(x)$ con una exactitud menor que ε.
while $(|b - a| \geq \varepsilon)$ **do**
  $m = \frac{a+b}{2}$;
  if $f(a)f(m) < 0$ **then**
    $b = m$;
  else
    $a = m$;
return Devuelve la raíz aproximada $m = \frac{a+b}{2}$;

Se pide programar el anterior algoritmo. Para ello se irá mejorando progresivamente sus prestaciones atendiendo al siguiente plan:

Versión1. Implementa el algoritmo para una función determinada $f(x) = \ln x$ y en un intervalo genérico $[a, b]$ y precisión ε. Debe devolver la raíz m.

Versión2. Añade que compruebe la hipótesis $f(a)f(b) < 0$, que devuelva los distintos intervalos obtenidos, puntos medios y valor de la fución en dichos puntos medios. La función Matlab también debe devolver la gráfica de la función para comprobar si es continua.

Versión3. Añade a la versión 2. que la función puede ser genérica. Testea el programa con los ejemplos $f_1(x) = x \ln x, f_2(x) = x - 2^{-x}, f_3(x) = e^{-x} + \sin x, f_4(x) = e^x - x^2 + 3x - 2$.

Utilización. ¿Qué interés máximo podría pagar un individuo que solicitase un préstamo de $C = 200000$ euros a 20 años y que pudiera pagar una hipoteca máxima de 1000 euros mensuales.

EJERCICIOS PROPUESTOS

Considérese los siguientes métodos para encontrar raíces de funciones:

1. Método de la Regula Falsi. El método de la Regula Falsi es idéntico al método de la bisección solo que en lugar de considerar el punto medio toma el punto de corte con el eje ox y el segmento que une los puntos $(a, f(a))$ y $(b, f(b))$. Esto es:

$$m = \frac{f(b)a - f(a)b}{f(b) - f(a)}$$

2. Método de punto fijo. Este método consiste en transformar el problema $f(x) = 0$ en uno equivalente del tipo $g(x) = x$ y a partir de ahí gener una sucesión del siguiente modo

$$\begin{cases} x_0 \\ x_{k+1} = g(x_k) \end{cases}$$

En caso de tratarse de una sucesión convergente $\lim x_k = x^*$ el límite x^* es una raíz de la función.

3. Método de Newton. Este es un caso particular de iteración funcional donde la función $g(x)$ está definida por

$$g(x) = x - \frac{f(x)}{f'(x)}$$

Lo que se trata es de averiguar computacionalmente cual de los anteriores métodos es más rápido. La forma de medir la velocidad de un método es mediante la expresión:

$$e_{k+1} \approx \alpha e_k^{\beta} \tag{1.3}$$

esta expresión indica que el error al efectuar $k+1$ iteraciones e_{k+1} es aproximademente proporcional a una potencia del error cometido al realizar k iteraciones. Cuando el número β es mayor el método es más rápido (¿Por qué?). Considérese el ejemplo concreto de calcular la raíz de $f(x) = \ln x$. Sabemos que su única raíz es $x^* = 1$ y por tanto podemos calcular en cada iteración k el error absoluto que se comente con los distintos métodos

$$e_k = |x_k - 1|$$

A partir de ellos se puede hacer un ajuste lineal (1.3) (buscar la función Matlab para esta tarea) para estimar los distintos α y β. ¿Qué valores se obtiene en cada método?

Formato .xls	Formato .csv	Formato .mat (MATLAB)

Cuadro 1.1: **Datos para la práctica 4 en diferentes formatos**

PRÁCTICA NÚMERO 4

Derivación e integración numérica

En la siguiente práctica se analizará con el programa Matlab y empleando los conceptos de derivada e integral unos datos médicos. Durante un largo periodo de tiempo se ha registrado la supervivencia de los pacientes a cierta intervención quirúrgica grave. Todos los pacientes intervenidos se monitorizaron durante el año siguiente a su intervención. De todos ellos, 68 fallecieron antes del primer año después de su intervención. La hoja Excel `datosmortalidad.xls` contiene el número de días que el paciente sobrevivió a la operación. La mortalidad de esta población se puede atribuir a factores naturales y a complicaciones derivadas de la operación. Se desea medir este segundo efecto. Lo primero de todo se debe formular matemáticamente el problema para poderse abordar de un modo exitoso. Supongamos que el tiempo de supervivencia, T, a partir de la operación es aleatorio. En matemáticas se denomina que T es una **variable aleatoria**. Esta variable tomará valores con una cierta probabilidad. La forma de medir esta probabilidad es a través de las denominadas funciones de densidad de probabilidad. Supongamos que la "ley" que gobierna esta variable aleatoria es una **función de densidad de probabilidad** $f(x)$ y la probabilidad que una persona fallezca en el intervalo $[a, b]$ viene dada por la expresión:

$$\mathbb{P}(a \leq T \leq b) = \int_a^b f(x)dx$$

El objetivo que perseguimos es analizar la función de densidad $f(x)$. El primer hecho matemático importante es la relación con la llamada **función de distribución**

$$F(t) = \mathbb{P}(T \leq t) = \int_0^t f(x)\mathrm{dx}$$

donde 0 está asociado al instante de la operación.

Por el teorema fundamental del Cálculo (suponiendo que $f(x)$ es continua) se tiene que

$$F'(t) = f(t) \tag{1.4}$$

A partir de los datos podemos construir una **función de distribución empírica** $\hat{F}(t)$ del siguiente modo. Supongamos que T_i es el instante (medido

en días en nuestros datos) donde fallece la $i-$ésima persona operada. Supongamos que $T_1 < T_2 < \cdots < T_{68}$ (en el archivo Excel están ordenados los datos). Entonces se tiene las siguientes imágenes:

$$\hat{F}(T_i) = \frac{i}{N}, \qquad i = 1, \cdots, 68 \tag{1.5}$$

donde N es el número de pacientes operados ($N = 247$ pacientes para nuestros datos).

En esta práctica queremos analizar visualmente el aspecto de la función de densidad empírica para detectar si existe algún efecto de la operación quirúrgica en la mortalidad. Para ello se realizan los siguientes pasos:

Paso 1: Cargar los datos del fichero Excel y almacenarlos en una variable `T`.

Paso 2: Dibujar la función de distribución empírica $F(x)$ definida por (1.5).

Paso 3: La relación (1.4) permite numéricamente calcular una aproximación a la función de densidad mediante aproximaciones laterales en los extremos y centrales en el resto:

$$\hat{f}(T_1) = \frac{\hat{F}(T_2) - \hat{F}(T_1)}{T_2 - T_1} = \frac{1/N}{T_2 - T_1} \tag{1.6}$$

$$\hat{f}(T_i) = \frac{\hat{F}(T_{i+1}) - \hat{F}(T_{i-1})}{(T_{i+1} - T_{i-1})} = \frac{2/N}{(T_{i+1} - T_{i-1})} \qquad i = 2, \cdots, 67 \tag{1.7}$$

$$\hat{f}(T_{68}) = \frac{\hat{F}(T_{68}) - \hat{F}(T_{67})}{T_{68} - T_{67}} = \frac{1/N}{T_{68} - T_{67}} \tag{1.8}$$

Paso 4: Alisamiento. Visualmente no se obseva ningún patrón. Esto es debido a que la función contiene mucho ruido. Una forma es filtrar los datos mediante las llamadas medias móviles de orden 3 o 5:

$$\tilde{f}_i^3 = \frac{\hat{f}_{i-i} + \hat{f}_i + \hat{f}_{i+1}}{3}$$

$$\tilde{f}_i^5 = \frac{\hat{f}_{i-2} + \hat{f}_{i-i} + \hat{f}_i + \hat{f}_{i+1} + \hat{f}_{i+2}}{5}$$

¿Qué se observa?

El fichero de la figura de la página siguiente contiene esta práctica.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PRACTICA4.M  Este script analiza la evolución de los pacientes
%              que han sufrido una operación de cadera
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T = xlsread('datosmortalidad.xls')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Dibujo de función de distribución empírica
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
F=[];       % Inicialización: función de distribución
fd=[];      % Inicialización: función de densidad de probabilidad
fdtilde3=[];% Alisamiento de medias móviles orden 3 de f.d.p.
fdtilde5=[];% Alisamiento de medias móviles orden 5 de f.d.p.

n=length(T);% Número de datos (pacientes fallecidos)
N=267;      % Número de pacientes operados
% Cálculo de función de distribución empírica
for i=1:n
    F(i)=(i-1)/N;
end
% Aproximación de la función de densidad: f(x)=F'(x)
for i=2:(n-1)
    fd(i)=(F(i+1)-F(i-1))/(T(i+1)-T(i-1));
end
fd(1)=(F(2)-F(1))/(T(2)-T(1));
fd(n)=(F(n)-F(n-1))/(T(n)-T(n-1));
% Representación de las funciones de distribución y densidad
subplot(2,2,1), plot(T,F);
xlabel('Día','FontSize',18,'FontWeight','bold','Color','r')
ylabel('Probabilidad','FontSize',18,'FontWeight',...
'bold','Color','r')
grid on
title('Funcion de distribución empírica')
subplot(2,2,3),plot(T,fd);
xlabel('Día','FontSize',18,'FontWeight','bold','Color','r')
grid on
title('Funcion de densidad empírica')
for i=2:(n-1) % A l i s a m i e n t o   o r d e n   3    %%%%%%
    fdtilde3(i)=(fd(i-1)+fd(i)+fd(i+1))/3;
end
fdtilde3(1)=(fd(1)+fd(2)+fd(3))/3;
fdtilde3(n)=(fd(n)+fd(n-1)+fd(n-2))/3;
for i=3:(n-2) % A l i s a m i e n t o   o r d e n   5    %%%%%%
    fdtilde5(i)=(fd(i-2)+fd(i-1)+fd(i)+fd(i+1)+fd(i+2))/5;
end
fdtilde5(1)=(fd(1)+fd(2)+fd(3)+fd(4)+fd(5))/5;
fdtilde5(2)=(fd(2)+fd(3)+fd(4)+fd(5)+fd(6))/5;
fdtilde5(n-1)=(fd(n-1)+fd(n-2)+fd(n-3)+fd(n-4)+fd(n-5))/5;
fdtilde5(n)=(fd(n)+fd(n-1)+fd(n-2)+fd(n-3)+fd(n-4))/5;
% Representación gráfica de las fdp alisadas
subplot(2,2,2),plot(T,fdtilde3);
xlabel('Día','FontSize',18,'FontWeight','bold','Color','r')
grid on
title('Funcion de densidad alisada orden 3')
subplot(2,2,4),plot(T,fdtilde5);
xlabel('Día','FontSize',18,'FontWeight','bold','Color','r')
grid on
title('Funcion de densidad alisada orden 5')
save datos.mat T fd fdtilde5; % Guarda variables T,fd,fdtilde5
                              % en el archivo datos.mat
% Los datos guardados se cargan con la sentencia: load datos.mat
```

Figura 1.15: **Practica4.m**

GUIÓN DE LA PRÁCTICA

1. La fórmula de los trapecios compuesta permite calcular el valor aproximado de una integral definida. Esta fórmula se expresa como:

$$\int_a^b f(x)\mathrm{dx} = \frac{h}{2}\left[f(a) + 2\left(\sum_{i=1}^{n-1} f(x_i)\right) + f(b)\right] \tag{1.9}$$

donde $h = \frac{b-a}{n}$ y $x_i = a + ih$ con $i = 1, \cdots, n-1$.

Desarrolla una función Matlab que permita calcular la integral de una función mediante el método de los trapecios compuesto para una función genérica y para un número de puntos arbitrario n.

La ecuación $x^2 + y^2 = 1$ define la gráfica de una circunferencia de radio 1 y centro el $(0, 0)$. Por tanto la integral

$$\int_{-1}^{1} \sqrt{1 - x^2}dx \tag{1.10}$$

representa el área de medio círculo de radio 1 cuyo valor es $\frac{\pi}{2}$. Emplea este ejemplo para testar la validez de la función Matlab desarrollada.

2. Si la función $f(x)$ es dos veces derivable en el intervalo $[a, b]$ y su derivada segunda $f''(x)$ es una función continua en $[a, b]$ el error de la fórmula de los trapecios compuesta se puede expresar por:

$$E_n = -\frac{f''(\theta)(b-a)^3}{12n^2} \qquad \theta \in (a, b) \tag{1.11}$$

A continuación vamos a comprobar numéricamente el error. Como conocemos el verdadero valor de la integral (1.10) numéricamente podemos calcular la fórmula del error. Lo haremos por pasos:

Paso 1: Calcular los errores absolutos cometidos con la fórmula de integración compuesta para los siguientes valores de n

$$|E_n| \qquad n = 1, \cdots, 100$$

Paso 2: Postularemos que la expresión del error es de la siguiente forma

$$|E_n| = \alpha n^{\beta}$$

La fórmula del error nos informa que $\beta = -2$ y que $\alpha = \left|-\frac{f''(\theta)(b-a)^3}{12}\right|$. Ahora vamos a ajustar los parámetros mediante mínimos cuadrados. Tomando $\ln$ en la anterior expresión obtenemos:

$$E_n' = \alpha' + \beta n' \tag{1.12}$$

donde $E'_n = \ln|E_n|$, $\alpha' = \ln\alpha$ y $n' = \ln n$. En el caso de nuestros datos la expresión (1.12) se puede escribir matricialmente

$$\begin{pmatrix} E'_1 \\ E'_2 \\ E'_3 \\ \cdots \\ E'_{100} \end{pmatrix} = \begin{pmatrix} 1 & \ln(1) \\ 1 & \ln(2) \\ 1 & \ln(3) \\ & \cdots \\ 1 & \ln(100) \end{pmatrix} \begin{pmatrix} \alpha' \\ \beta \end{pmatrix}$$

De forma abreviada el anterior sistema de ecuaciones se puede escribir en forma abreviada $b = Ax$, donde b es el vector de errores, A la matriz de datos y x el vector de incógnitas (el valor de los parámetros).

La sentencia `x=A\b` resuelve el anterior sistema. Si no tiene solución, como es el caso, busca una estimación mínimo cuadrática para encontrar una solución que mejor ajuste (ver en ejercicio propuesto el método de estimación mínimo cuadrático).

Paso 3: Una vez calculado el β empírico, ¿Ajusta bien la fórmula teórica con la observación experimental? Representa gráficamente ambos modelos.

EJERCICIOS PROPUESTOS

1. En el siguiente ejercicio se pide repetir la práctica anterior pero empleando la fórmula de integración compuesta de Simpson.

$$\int_a^b f(x) \approx \frac{h}{3}\left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots 4f(x_{n-1}) + f(x_n)\right)$$

donde $h = \frac{b-a}{n}$.

La fórmula del error tiene la expresión

$$-\frac{f^{iv)}(\theta)(b-a)^5}{180n^x} \qquad \theta \in (a,b)$$

A partir de los resultados numéricos postula un valor de x para la fórmula anterior.

2. Ajuste mínimo cuadrático. de una función del tipo $f(x) = K + \beta exp(-\alpha x)$

Formato .xls	Formato .csv	Formato .mat (MATLAB)

Cuadro 1.2: **Datos para la práctica 5 en diferentes formatos**

PRÁCTICA NÚMERO 5

Modelización Matemática I

Un modelo matemático es una *imagen teórica* de la realidad de una parcela del "mundo real" en la que sus elementos se reemplazan por entes abstractos y relaciones entre estos. Con facilidad se confunde "modelo" y "realidad", al considerar que el modelo es una síntesis de la "verdad"del mundo. Una definición clarificadora de lo que es un modelo es la dada por Minsky:

> "Un objeto M, es un modelo de un objeto R, para un observador O cuando M contesta a las preguntas que formula O sobre R".

En esa definición se pone de manifiesto que el objetivo de un modelo es el de obtener una mejor comprensión de la realidad mediante la contestación de las preguntas del observador. Lo que se exige a un modelo es que sea consistente y concordante con las observaciones empíricas, pero no que los mecanismos establecidos en el modelo sean los que realemente operan en la realidad. Y es precisamente la coincidencia del resultado final, predicciones con observaciones, lo que produce esta confusión entre "realidad" y "modelo".

Cabe enfatizar que la elaboración de modelos matemáticos es un enfoque científico para estudiar sistemas complejos. Un esquema simplificado del proceso de modelación, podría tener las siguientes etapas:

1. *Elaboración de un modelo matemático,* definiendo los elementos fundamentales del sistema y sus interrelaciones. En esta fase se aplica un *principio de parsimonia,* bajo el cual se dabe dar un equilibrio entre la relevancia de un cierto elemento y la complejidad del modelo resultante.

2. *Validación del modelo,* evaluando las hipótesis sobre las que se ha construido el model y analizando la *sensibilidad* de sus elementos. Se compara las predicciones y comportamiento del modelo con la información experimental disponible y, en caso necesario, se construyen modelos alternativos.

3. *Análisis del modelo,* mediante razonamientos teóricos y computacionales sobre estos modelos, que en ocasiones, conducen a "conclusiones" que son trasladables al mundo físico. En esta fase se requiere de algoritmos matemáticos para resolver el modelo.

4. *Utilización del modelo.* Aplicar las conclusiones obtenidas.

La figura 1.16 muestra esquemáticamente los pasos anteriormente descritos:

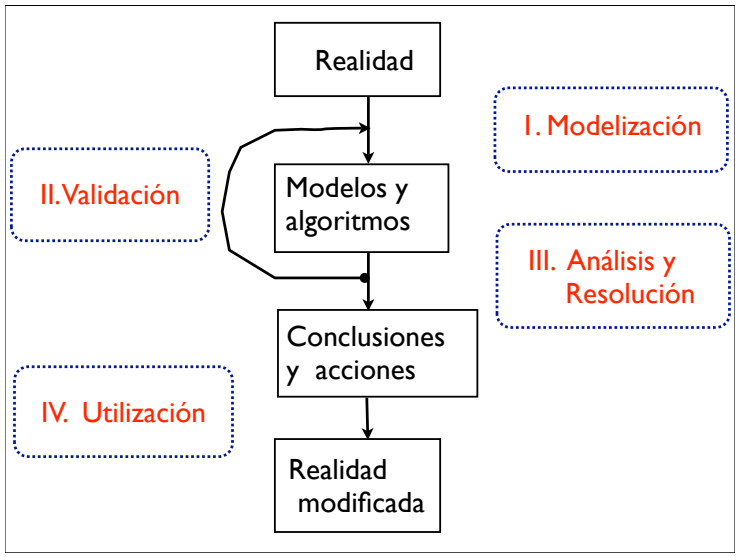

Figura 1.16: Etapas de la modelización matemática

GUIÓN DE LA PRÁCTICA

Las ecuaciones diferenciales son modelos matemáticos ampliamente empleados en la ciencia y en la técnica. Estos modelos representan implícitamente una función (solución) a través de un conjunto de relaciones que satisface la propia función y sus derivadas. El ejemplo más sencillo lo constituyen las ecuación diferencial de primer orden:

$$\begin{cases} y'(x) = r(x, y) \\ y(x_0) = y_0 \end{cases}$$

En esta práctica se desea obtener un modelo matemático de la evolución (en el tiempo) del grado de implantación de cierta especie de pinos en el lago Michigan. La forma de medir este fenómeno es a través de la proporción del área ocupada en cada instante t, que se denotará $P(t)$. Supóngase que el modelo propuesto es:

Considérese la siguiente ecuación diferencial

$$P'(t) = kP(t) - a[P(t)]^2 \qquad (1.13)$$

donde

- t instante de tiempo
- $P(t)$ es el porcentaje en el instante t de una determinada especie
- $P'(t)$ es la derivada de la anterior función que informa de su variación en el tiempo
- $kP(t)$ es un término asociado al crecimiento (expansión)
- $-aP(t)^2$ este término está asocidado al decrecimiento de la especie.

Este modelo depende de dos parámetros k y a que controlan el nivel de expansión y contracción de una especie. El objetivo perseguido en esta práctica es encontrar el mejor vector de parámetros (a, k) que permitan reproducir los datos observados. Según observaciones de polen fósil encontrado se ha determinado que el nivel de implantación de dos especies de pino, que denotamos con los subíndices 1 y 2, han sido el mostrado en la tabla 1.

Fuente: http://www.geom.uiuc.edu/education/calc-init/population/logistic.html

Cuadro 1.3: Evolución de la implantación de dos especies de pino en el lago Michigan

Miles de Años (atrás)	t	$P_1(t)$	$P_2(t)$
9131	0.0	53.4	3.2
8872	0.259	65.5	0.0
8491	0.640	61.8	3.7
8121	1.010	55.2	3.4
7721	1.410	60.4	1.7
7362	1.769	59.4	1.8
7005	2.126	50.6	10.6
6699	2.432	51.6	7.0
6444	2.687	40.0	21.2
5983	3.148	29.7	34.2
5513	3.618	25.0	40.4
5022	4.109	32.5	29.8
4518	4.613	22.7	46.2
4102	5.029	31.6	33.0
3624	5.507	32.5	37.6
3168	5.963	27.1	39.5

Hito 1: Visualización de datos. Los datos mostrados en la tabla anterior se pueden encontrar en el archivo EvolucionPinosMichigan.xls proporcionado. Para cargar la información contenida en este se utilizará la función xlsread que proporciona MATLAB, indicándole donde se encuentra el archivo, teniendo en cuenta que si se encuentra dentro de la carpeta de MATLAB o en la que estemos ejecutando el código solo será necesario poner su nombre, pero si se encuentra en otro diferente habrá que indicar la ruta completa:

```
»datos=xlsread('EvolucionPinosMichigan.xls');
```

Con esto se dispondrá de toda la información almacenada en la matriz datos. Para comprobar que se han cargado los datos en la matriz datos, vamos a representar la evolución de los dos tipos de pinos

```
»plot(datos(:,2),datos(:,3),'o-',datos(:,2),datos(:,4),'*-');

»xlabel Tiempo, ylabel 'Tipo de Pinos';
```

Hito 2: Definición implícita de la solución. El objetivo que se persigue es ilustrar el proceso de modelización matemática y en concreto la fase de validación. Las ecuaciones diferenciales es un pretexto para ilustrar esta fase. Primeramente resolveremos la ecuación diferencial. Dividimos ambos lados de (1.13) por $kP(t) - aP^2(t)$

$$\frac{P'(t)}{kP(t) - aP^2(t)} = 1 \Rightarrow \frac{P'(t)}{(k - aP(t))P(t)} = P'(t)\left(\frac{\frac{a}{k}}{aP(t) - k} - \frac{\frac{1}{k}}{P(t)}\right) = 1$$

Integrando ambos lados respecto a t

$$\int \frac{\frac{a}{k}P'(t)}{aP(t)-k}\mathrm{dt} - \int \frac{\frac{1}{k}P'(t)}{P(t)}\mathrm{dt} = \int 1\mathrm{dt} \Rightarrow$$

$$\frac{1}{k}\left(\ln|aP(t)-k| - \ln|P(t)|\right) = t + C \qquad (1.14)$$

donde C es una constante de integración. El cálculo de la constante C se puede calcular a partir de un punto de la función. Por ejemplo, si en el instante inicial $t = 0$ había $P(0) = P_0$ proporción de pinos entonces ese valor debe cumplir la anterior solución y por tanto

$$\frac{1}{k}\left(\ln|aP(0)-k| - \ln|P(0)|\right) = 0 + C \Rightarrow C = \frac{1}{k}\left(\ln|aP_0-k| - \ln|P_0|\right)$$

La solución (1.14) define $P(t)$ implícitamente, esto es, no se puede despejar el valor de $P(t)$ en función de t. Por ejemplo, para calcular la imagen de $P(3)$, tenemos que sustituir $t = 3$ y resolver la ecuación en la incógnita $P(3)$.

MATLAB tiene implementado una función para calcular raíces. Supongamos que queremos encontrar las raíces de

$$f(x) = x^2 - 4x + 1.$$

Primero definimos una función que depende de la variable x:

```
»f=@(x) x*x-4*x+1
```

La función fzero calcula la raíz de una función. Como parámetros de entrada tiene una función y un punto donde iniciar el método numérico. En el siguiente ejemplo consideramos la función $f(x) = x^2 - 4x + 1$ y tomamos como punto inicial el 0.

```
»fzero(f,0)
```

El anterior esquema también es aplicable a nuestro problema. Primero se define una función auxiliar que define la función (1.14) en función de la variable P y de todos los parámetros considerados. Posteriormente se fijan los valores de los parámetros y a traves del cálculo de las raíces de una función (que equivale a la solución de una ecuación) se obtiene el valor de la imagen. La función implicita.m implementa una función MATLAB que calcula las imagenes $P(t)$ en un vector de puntos $tt = (t_1, t_2, \cdots, t_n)$ dado los parámetros a, k, $P(0)$ y el propio vector tt.

Aplica la función implicita.m al vector de tiempos t dado en la tabla 1.3 para los siguientes casos:

1) $a = 0.05$ $k = 0.2$ $P(0) = 0.534$
2) $a = 0.005$ $k = 0.2$ $P(0) = 0.534$
3) $a = 0.005$ $k = 0.1$ $P(0) = 0.534$

```
function P=implicita(t,a,k,P0)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% implicita calcula las imágenes de una función implícita
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) N o m b r e:
%       implicita
% (2) D a t o s    d e   e n t r a d a:
%           t vector de instantes de tiempo t=[t1,t2,t3,..,tn]
%           a parámetro a de la ecuación diferencial
%           k parámetro k de la ecuación diferencial
%           P0 proporcion inicial de pinos en el instante t=0
%(3) D a t o s    d e   s a l i d a:
%           P es un vector [P(t1) P(t2),...P(tn)] conteniendo
%           la evolucion de las proporciones de los pinos
%(4) P r o c e d i m i e n t o   d e   c á l c u l o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculamos el valor de la constante C en funcion de P0
C=1/k*(log(abs(a*P0-k))-log(abs(P0)));
% El bucle siguiente  calcula el vector de imágenes
for i=1:length(t)
    f=@(P) 1/k*(log(abs(a*P-k))-log(abs(P)))-C-t(i);
    P(i)=fzero(f,0.5); % resolución método Newton ecuación f(P)=0
end
end
```

Figura 1.17: **implicita.m**

2. Representación gráfica de la solución. Construir dos gráficas, uno para cada tipo de pino, que represente simúltaneamente las observaciones y la evolución predicha por el modelo en los tres casos anteriores. Notar que los datos se miden en porcentajes mientras que $P(t)$ en proporciones.

3. Un índice de la bondad de ajuste. Las gráficas anteriores permiten visualmente comprobar el ajuste de los modelos teóricos a los datos observados pero es difícil decidir cual de ellas es la más adecuada. Para ello deberíamos disponer de algún índice que nos permitiera comparar los diferentes ajustes. El índice más usual es la suma de los errores al cuadrado entre los valores observados y los predichos por el modelo. Más formalmente, supóngase que $P(t_j, a, k, P_0)$ es la solución de la ecuación diferencial para el valor t_j empleando los parámetros a, k y P_0. Además, supongamos que se han observado los siguientes datos del modelo $P_{\text{obs}}(t_j)$ con $j = 1, \ldots, n$. La suma de los errores al cuadrado se define:

$$H(a, k, P_0) = \sum_{j=1}^{n} \left(P(t_j, a, k, P_0) - P_{\text{obs}}(t_j)\right)^2 .$$

El ajuste será mejor cuanto menor sea el valor de $H(a, k, P_0)$.

En nuestro problema los datos de la función $P(t_j)$ están definidos en la tabla 1 y almacenados en la variable datos. Por otro lado

```
function H=SumaErrores(t,Pobs,a,k,P0)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SumaErrores calcula la suma de errores al cuadrado entre
% la solución de la ecuación diferencial y los datos observados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) N o m b r e:
%       SumaErrores
% (2) D a t o s    d e   e n t r a d a:
%           t vector de instantes de tiempo t=[t1,t2,t3,..,tn]
%           Pobs propoción de pinos observados en los instantes t
%           a parámetro a de la ecuación diferencial
%           k parámetro k de la ecuación diferencial
%           P0 proporcion inicial de pinos en el instante t=0
%(3) D a t o s    d e   s a l i d a:
%           H  suma de errores al cuadrado
%(4) P r o c e d i m i e n t o   d e   c á l c u l o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P=implicita(t,a,k,P0);  % Resolución ecuación diferencial
P=100*P;          % Conversión de las proporciones a porcentajes
H=0;              % Inicialización de variable suma de errores
for i=1:length(t)
    H=H+(P(i)-Pobs(i)).^2;
end
end
```

Figura 1.18: **SumaErrores.m**

la función implicita.m nos permite calcular los valores estimados $P(t_j, a, k, P_0)$. El M-fichero SumaErrores.m implementa la función $H(a, k, P_0)$. Aplicar la función SumaErrores.m para determinar cual de los tres anteriores conjuntos de parámetros es el más adecuado.

4. El problema de ajuste mínimo cuadrático. Es necesario diseñar procedimientos que obtengan automáticamente los valores de los parámetros de un modelo matemático. Uno de los métodos más aplicados es el *ajuste mínimo cuadrático* que consiste en estimar los parámetros (a^*, k^*, P_0^*) de la ecuación diferencial que minimizan la suma de los errores al cuadrado. Formalmente

$$H(a^*, k^*, P_0) = \text{Minimizar}_{\ a,k,P_0} H(a, k, P_0) \tag{1.15}$$

El problema de optimizar una función de varias variables es un problema difícil. MATLAB dispone de algunas funciones que implementas ciertos métodos de optimización. Una de estas funciones es fminsearch que codifica el método de Nealder-Mead. Vamos a ilustrar su uso. Si quisieramos calcular el mínimo de la siguiente función

$$f(x_1, x_2) = x_1^2 + x_2^2 + 2 * x_1 x_2$$

primeramente introduciríamos la función:

```
»f= @(x) x(1)*x(1)+x(2)*x(2)+2*x(1)*x(2)
```

Esta expresión indica que f depende de un vector x. En la definición de las operaciones a realizar se pone de manifiesto que el vector x tiene dos componentes, la componente $x(1)$ y $x(2)$. Si quisieramos evaluar la función en el punto $(1,2)$ tendríamos que pasarle como parámetro de entrada el vector $[1,2]$ y escribir

```
»f([1,2])
```

Si hubieramos escrito $f(1,2)$ Matlab entendería que le estamos pasando dos argumentos y mostraría un error.

La función fminsearch(NombreFuncion,PuntoInicial) tiene como argumentos de entrada el nombre de la función, NombreFuncion, y el vector inicial PuntoInicial. En nuestro caso como la función depende de dos variables el vector inicial tiene que tener dos componentes en las que se almacena el valor inicial para cada una de las variables. Si queremos minimizar la función f comenzando el algoritmo en el punto $x = [1,1]$ escribiríamos:

```
»[xopt,fval]=fminsearch(f,[1,1])
```

Al ejecutar esta función obtenemos que xopt es un mínimo (relativo) y el valor de la función objetivo fval=f(xopt).

Los procedimientos de optimización pueden ser no convergentes o consumir un alto tiempo de cómputo por este motivo se debe especificar un criterio de terminación diferente para cada tipo de problemas. La sentencia

```
»opciones=optimset('MaxFunEvals',5000,'MaxIter',500,'TolX',1e-3
)
```

modifica los criterios de paro. El número máximo de veces que va a evaluar la función objetivo es 5000, el máximo número de iteraciones a realiar son 500 y si la distancia entre dos soluciones consecutivas es menor que 10^{-3} termina. Además solo muestra por pantalla el resultado final. Si queremos ejecutar el procedimiento con estas nuevas opciones escribimos:

```
»[x,fval]=fminsearch(f,[1,1],opciones)
```

El script sesion5.m contiene los diferentes hitos anteriores y la resolución del problema (1.15) para ajustar los parámetros del modelo.

```
%%%%%%%%%%%%%%%%%%%%%%%%    H i t o   1     %%%%%%%%%%%%%%%%%%%%
datos=xlsread('EvolucionPinosMichigan.xls');
plot(datos(:,2),datos(:,3),'o--g',datos(:,2),datos(:,4),'*--r');
xlabel 'Tiempo', ylabel 'Porcentaje de Pinos';
%%%%%%%%%%%%%%%%%%%%%%%%    H i t o   2     %%%%%%%%%%%%%%%%%%%%
t=datos(:,2);
Pobs=datos(:,3);        % Datos evolución Pino 1
hold on
%Caso 1
a=0.05; k=0.2; P0=0.534;
P=implicita(t,a,k,P0); % Evolución pinos para el Caso 1
P=100*P;               % Conversión proporciones a porcentajes
plot(t,P,'<-');
%Caso 2
a=0.005; k=0.2; P0=0.534;
P=implicita(t,a,k,P0);% Evolución pinos para el Caso 2
P=100*P;
plot(t,P,'>-');
%Caso 3
a=0.005; k=0.1; P0=0.534;
P=implicita(t,a,k,P0); % Evolución pinos para el Caso 1
P=100*P;
plot(t,P,'s-');
%%%%%%%%%%%%%%%%%%%%%%%%    H i t o   3     %%%%%%%%%%%%%%%%%%%%
a=0.05; k=0.2; P0=0.534;        % Caso 1
H1=SumaErrores(t,Pobs,a,k,P0)
a=0.005; k=0.2; P0=0.534;       % Caso 2
H2=SumaErrores(t,Pobs,a,k,P0)
a=0.005; k=0.1; P0=0.534;       % Caso 3
H3=SumaErrores(t,Pobs,a,k,P0)
%%%%%%%%%%%%%%%%%%%%%%%%    H i t o   4     %%%%%%%%%%%%%%%%%%%%
H=@(x) SumaErrores(t,Pobs,x(1),x(2),x(3)); % Criterio de ajuste
x1=[a,k,P0]; % Parámetros iniciales
[par_opt,fopt]=fminsearch(H,x1); % Optimización criterio de ajuste
% Representación de la solución encontrada
P=implicita(t,par_opt(1),par_opt(2),par_opt(3));
P=100*P;
plot(t,P,'*-k','LineWidth',2);
```

Figura 1.19: **practica5.m**

EJERCICIOS PROPUESTOS

1. Algoritmo de optimización del gradiente. Considerése que se desea calcular el mínimo de una función $f(x)$. Muchos procedimientos definen una sucesión $\{x_k\}$ de modo que el límite de dicha sucesión $x^* = \lim_k x_k$ es un mínimo (local) de dicha función. Uno de estos métodos se le conoce con el nombre del método del gradiente o del mayor descenso. Ahora vamos a introducir este método. Supóngase que en el punto x_k la derivada $f'(x_k) < 0$, supóngase que el siguiente valor que queremos tomar tiene la forma $x_{k+1} = x_k + \delta$ ¿Conviene tomar valores positivos o negativo de δ?>Y si el valor de la derivada $f'(x_k) > 0$ fuese negativo? Lo anterior motiva construir una sucesión del siguiente modo

$$x_{k+1} = x_k - cf'(x_k)$$

donde el parámetro c mide cuanto nos movemos en la dirección indicada por la derivada. en muchos ejemplos prácticos dicho parámetro se toma de la iteración y se elige $c = \frac{1}{k}$. Aplicar el anterior método para optimizar la función $f(x) = x^2$

2. Caso multidimensional. El anterior algoritmos también se puede aplicar a funciones de varias variables. En este caso el papel de la derivada lo desempeña el gradiente. Para ilustrar este caso consideremos una función $f(x_1, x_2) = x_1^2 + 2x_1x_2 + 3x_2^2$ que depende de las variables x_1 y la variable x_2. El gradiente se define:

$$\bigtriangledown f(x_1, x_2) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix}$$

Las derivadas parciales de una función f respecto a una variable se calculan considerando el resto de variables como constantes y derivando respecto a la variable en cuestión. En nuestro ejemplo obtendríamos:

$$\bigtriangledown f(x_1, x_2) = \begin{pmatrix} 2x_1 + 2x_2 \\ 3x_2 + 2x_1 \frac{\partial f}{\partial x_2} \end{pmatrix}$$

Finalmente obtendríamos la fórmula iterativa:

$$x_{k+1} = x_k - c \bigtriangledown f(x_k)$$

donde x_k, x_{k+1} y $\bigtriangledown f(x_k)$ son vectores. Se pide:

   a) Calcula las curvas de nivel de $f(x_1, x_2) = x_1^2 + 2x_1x_2 + 3x_2^2$ para localizar visualmente el mínimo.

   b) Aplica este método para aproximar el mínimo de la función. Haz varias comprobaciones para diferentes valores de c y puntos iniciales. Puedes dibujar sobre la curva de nivel el camino seguido por el algoritmo.

c) Aplica el anterior método al ejemplo de ajuste de parámetro de la práctica. En este caso tienes que emplear métodos de derivación numérica para aproximar las derivadas parciales ya que no se puede derivar analíticamente.

Formato .xls	Formato .csv	Formato .mat (MATLAB)

Cuadro 1.4: **Datos para la Archivo 2 (práctica 6) en diferentes formatos**

Formato .xls	Formato .csv	Formato .mat (MATLAB)

Cuadro 1.5: **Datos para la Archivo 1(práctica 6) en diferentes formatos**

PRÁCTICA NÚMERO 6

Ejemplo de examen

Esta práctica consta de dos ejercicios propuetos en examenes en cursos anteriores, uno en la propia práctica y otro en ejercicio propuesto. En esta práctica ya no se requiere de nuevas instrucciones de MATLAB sino de poseer las destrezas necesarias para poder aplicar los contenidos de las sesiones anteriores al cálculo de fórmulas matemáticas. Para realizar exitósamente esta práctica se require enteder lo que se desea calcular y posteriormente saber cómo se puede implementar en un lenguaje de programación, en particular en MATLAB.

GUIÓN DE LA PRÁCTICA

El sistema que utiliza Google para la ordenación de las páginas webs cuando realizamos una búsqueda determinada es el `PageRank`. Este algoritmo se utiliza para asignar de forma numérica la relevancia de las páginas web indexadas por el motor de búsqueda.

Para ello, Google interpreta un enlace de una página A a una página B como un voto, de la página A, a la página B teniendo en cuenta la página que emite el voto, ya que los votos emitidos por las páginas con un `PageRank` elevado valen más. Por lo tanto, el `PageRank` de una página refleja la importancia de la misma según Google en Internet.

1. Considérese el archivo archivo2.xls, en el que se dan todos los datos necesarios para calcular el `PageRank` un determinado mes. En cada una de las filas se representa la información concerniente a una página que enlaza nuestra web, representando la primera columna el `PageRank` de esa página y la segunda el número de enlaces salientes a otras webs incluyendo la nuestra. Con esta información podemos calcular el `PageRank` de nuestra web mediante la siguiente fórmula:

$$\text{PageRank} = (1-d) + d * \sum_{i=1}^{n} \frac{PR(i)}{C(i)},$$

donde:

- PageRank es el `PageRank` de nuestra página A.
- d es un factor de amortiguación que tiene un valor entre 0 y 1.
- $PR(i)$ son los valores de `PageRank` que tienen cada una de las páginas i que enlazan a A.
- $C(i)$ es el número total de enlaces salientes de la página i (sean o no hacia A).

Cargar el archivo2.xls y calcular el `PageRank` de nuestra web en este mes tomando $d = 0,85$.

2. Queremos predecir el `PageRank` para el próximo mes (80) a partir de los datos que ya conocemos. Un modo sencillo es mediante lo que se denomina como *recta de regresión lineal simple*. Este método lo que hace es ajustar la evolución de los datos a la recta:

$$y_i = \alpha + \beta x_i, \qquad i = 1, \cdots, n$$

donde x_i es el i-ésimo mes e y_i es su `PageRank`.

En el fichero archivo1.xls se muestra la evolución del `PageRank` de nuestra página web (2ª columna, la y_i) a lo largo de una serie de meses (1ª columna, la x_i). Se pide:

a) Cargar el archivo1.xls en la variable evolucion.

b) Calcular los valores medios de las variables:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1.16}$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{1.17}$$

c) Utilizando el método de mínimos cuadrados, el valor de los parámetros vienen definidos por:

$$\beta = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\alpha = \bar{y} - \beta\bar{x}$$

Calcular estos parámetros.

d) Obtener el valor estimado del `PageRank` de nuestra página para el próximo mes.

e) Representar gráficamente la evolución del `PageRank` de nuestra página y la recta de regresión calculada.

EJERCICIOS PROPUESTOS

1. Sea $f(x)$ una función definida en el intevalo $[a,b]$. Una aproximación de la integral definida de $f(x)$ en el intervalo $[a,b]$ viene dada por

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[f(x_0)+f(x_n)+2\sum_{i=1}^{n-1} f(x_i)\right] \tag{1.18}$$

donde $x_i = a+ih$ para $i = 0,1,\cdots,n$ y $h = (b-a)/n$.

a) Definir una función MATLAB que permita calcular la expresión (1.18) a partir de los datos de entrada: i) $f(x)$, ii) a, iii) b y iv) n.

b) Considérese las siguientes funciones:

$$x(t) = \int_0^t \cos(u^2)du \tag{1.19}$$

$$y(t) = \int_0^t \text{sen}(u^2)du \tag{1.20}$$

Aplicar la función desarrollada en el apartado 2.a) para dar un valor aproximado de $x(1)$ y de $y(1)$, tomando $n = 100$.

c) Considérese la partición $t_0 < t_1 < \cdots < t_{1000}$ de puntos uniformemente espaciados del intervalo $[-4\pi, 4\pi]$. Realizar un M-fichero para representar gráficamente $[x(t_i), y(t_i)]$ desde $i = 0,\cdots,1000$. Aplicar la función desarrollada en el apartado 2.a) para el cálculo de las integrales y tomar el valor $n = 100$.

d) ¿Qué ocurre si se repite el apartado anterior tomando $n = 10$ en el cálculo de las integrales? Emplear las propiedades de la integral definida para optimizar los cálculos del M-fichero realizado en el apartado anterior de modo que calculando las integrales con $n = 10$ se obtenga una gráfica similar al apartado anterior. Ayuda: i) ¿Qué relación existe entre $x(t_i)$ y $x(-t_i)$?, ii) ¿Qué relación existe entre $x(t_i)$ y $x(t_{i+1})$? y iii) plantear las mismas relaciones para calcular $y(t_i)$.

mam.pgm	imagen.png

Cuadro 1.6: **Datos para la práctica** 7

PRÁCTICA NÚMERO 7

Visualización 2D

.

Hoy en día la mejor manera disponible para poder enfrentarnos al cáncer de mama es que este sea diagnosticado lo más precozmente posible ya que un cáncer detectado a tiempo es un cáncer potencialmente curable. El método de detección precoz más eficaz para el cáncer de mama es la mamografía, un examen radiológico de la mama que permite obtener imágenes muy claras de la estructura interna de esta. Por ello el tratamiento de imagen mediante la utilización de diferentes filtros que la modifiquen es una herramienta muy útil en la medicina moderna para la detección automática de las conocidas como regiones de interés, es decir, regiones que pueden indicar la posible existencia de un cáncer.

Para la realización de este examen se proporciona el archivo **mam.pgm** mostrado en la Figura 1.20, el cual contiene una imagen de una mamografía real. Esta imagen está representada en escala de grises, es decir, cada uno de los píxeles (puntos que conforman la imagen) tiene un color que puede ir desde el negro al blanco, pasando por las diferentes tonalidades de gris. Cada uno de estos colores se define mediante un número entero que va desde el 0 (color negro) al 255 (color blanco), por lo que la imagen se puede representar como una matriz de números enteros, en la que cada celda de la matriz se correspondería con cada uno de los píxeles de la imagen y tendría almacenado su color.

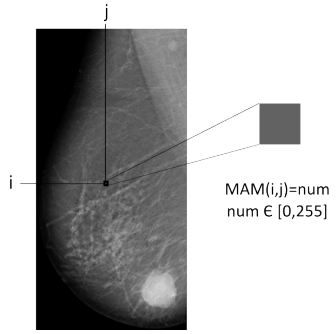

Figura 1.20: Visualización del archivo mam.pgm

GUIÓN DE LA PRÁCTICA

1. Cargar el archivo **mam.pgm** en la matriz `MAM` utilizando la función **imread** de MATLAB. Nota: la función **imread** funciona igual que **xlsread**.

2. El histograma de una imagen representa el número de apariciones de cada uno de los valores de gris en la imagen, es decir, cuantos píxeles existen en la imagen de cada color en concreto. Calcular el histograma de la imagen **mam.pgm** utilizando la matriz `MAM` del apartado anterior siguiendo los siguientes pasos:

   a) Generar un vector `XHist` que contenga todos los colores desde el negro (0) al blanco (255) representados numéricamente.

   b) Calcular el vector `YHist`, almacenando en cada una de sus componentes el número de veces que aparece cada color en la matriz `MAM`.

   c) Representar en una figura los dos vectores anteriores.

3. Uno de los filtros utilizados para el tratamiento de imagen es el conocido como umbralizado, el cual dada la representación numérica C de un color, modifica la imagen cambiando por color negro todos los píxeles cuyo valor esté por debajo de C y por blanco todos los píxeles cuyo valor sea igual o mayor que C.

   a) Umbralizar la imagen almacenada en `MAM` con $C = 175$ y almacenarla en `MAMUmb`.

   b) Representar en una misma figura la imagen contenida en `MAM` y la contenida en `MAMUmb` utilizando la función **imshow** de MATLAB.

EJERCICIOS PROPUESTOS

1. Para la realización de este ejercicio se proporciona el archivo **imagen.png** mostrado en la Figura 1.21, el cual contiene una imagen en formato .png. Esta imagen está representada en color. Cada uno de los píxeles (puntos que conforman la imagen) tiene asociado tres componentes que define la intensidad de rojo (R), verde (G) y azul (B) del pixel. Cada uno de estos valores se define mediante un número entero que va desde el 0 (ausencia de color) al 255 (máximo color), por lo que una imagen se puede representar con tres matrices de números enteros, en la que cada celda se correspondería con cada uno de los píxeles de la imagen y tendría almacenado la intensidad de su color respectivo. Por ejemplo, el color blanco es la unión de todos los colores y se define poniendo los 3 valores de un pixel a 255. El color negro es la ausencia de color y se define poniendo los tres valores de un pixel a 0. Con otras combinaciones se formarían los demás colores.

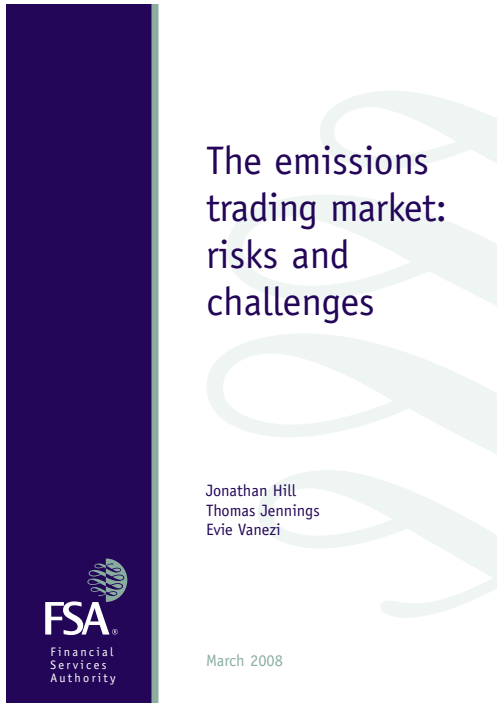

Figura 1.21: Visualización del archivo Imagen.png

a) Cargar el archivo **imagen.png** en la matriz A utilizando la función **imread** de MATLAB y dibujala con la instrucción **imshow**.

Notas:

La función **imread** funciona igual que **xlsread**.

`A(i,j,1)` representa la intensidad de rojo del pixel de la fila i y columna j;
`A(i,j,2)` representa la intensidad de verde del pixel de la fila i y columna j;
`A(i,j,3)` representa la intensidad de azul del pixel de la fila i y columna j.

b) Construye una matriz `ANegra` que represente la figura en blanco y negro. Esto es, los pixeles que tenga un color diferente al blanco $(255, 255, 255)$ debe convertirse a negro $(0, 0, 0)$. Representa la figura `ANegra`.

c) Calcula la proporción de negro que hay en la imagen definida por `ANegra`.

d) Dibuja la imagen `A` reflejada en un espejo.

PRÁCTICA NÚMERO 8

Optimización

$\mathbb{R}^3$ es el conjunto de puntos de coordenadas (x, y, z). En $\mathbb{R}^3$ podemos considerar superficies y éstas se pueden representar por los puntos de la forma $(x, y, z(z, y))$ al hacer variar $(x, y) \in D$. Por ejemplo podemos considerar la siguiente media esfera, definida por

$$\left(x, y, \sqrt{4 - x^2 - y^2}\right) \qquad (x, y) \in D \tag{1.21}$$

donde $D := \{(x, y) : x^2 + y^2 \leq 4\}$ define el círculo de centro $(0, 0)$ y radio 2

Con las siguientes instrucciones se representa la superficie: `x=-2:.1:2; y=-2:.1:2; [X, Y]=meshgrid(x,y);` $Z = sqrt(4 - X.^2 - Y.^2)$`; mesh(Z)`

El problema que nos ocupa en esta práctica es encontrar el camino más corto dentro de la superficie entre dos puntos dados. Por ejemplo si queremos ir en avión desde una ciudad a otra >cual camino emplearíamos? Este problema equivale a encontrar el camino más corto entre dos puntos dados en una esfera.

Para formular este problema necesitamos los siguientes resultados:

Una curva paramétrica viene caracterizada por la siguiente expresión:

$$P(t) = (x(t), y(t), z(t)) \qquad t \in [a, b]$$

donde t es un parámetro y $P(t)$ es un punto de la curva. Cuando t recorre todos los puntos del intervalo $[a, b]$, $P(t)$ recorre todos los puntos de la curva.

Si quisieramos calcular la longitud de dicha curva tendríamos que calcular la siguiente integral

$$\int_a^b \|P'(t)\| = \int_a^b \sqrt{x'(t)^2 + y'(t)^2 + z'(t)}\mathrm{dt} \tag{1.22}$$

donde $x'(t), y'(t), z'(t)$ son las derivadas de las funciones coordenadas respecto al parámetro t.

Se consideran los siguientes puntos de la esfera $P_1 = (-1, 1, \sqrt{2})$ y $P_2 = (1, 1, \sqrt{2})$. Se desea ir desde P_1 hasta P_2 a través de las siguientes curvas:

$$x_a(t) = t \tag{1.23}$$

$$y_a(t) = a(t - 1)(t + 1) + 1 \tag{1.24}$$

$$z_a(t) = \sqrt{4 - t^2 - [a(t - 1)(t + 1) + 1]^2} \tag{1.25}$$

donde a es un parámetro y $t \in [-1, 1]$.

Observación. Para interpretar geométricamente este problema el modo en el que se construyeb estas curvas es el siguiente. Consideramos el plano $x - y$, en él las proyecciones $\hat{P}_1 = (-1, 1)$ y $\hat{P}_2 = (1, 1)$ de los

puntos P_1 y P_2. Unimos estas proyecciones por las parábolas $y(x) = a(x-1)(x+1)+1$ y finalmente consideramos las curvas sobre la superficie $(x, y(x), \sqrt{4-x^2-y^2(x)}$. Tomamos como parámetro $x = t$

Se pide:

1. Construir una función Matlab que permita dibujar la curva sobre la superficie para un valor determinado de a. **Ayuda:** Función `plot3(x,y,z)`
2. ¿Cual es el valor mayor y menor de a para el que está definida estas curvas?
3. Construye una función Matlab para calcular la longitud de la curva en función de a.
4. Calcula la curva de mínima longitud que une los puntos P_1 y P_2

GUIÓN DE LA PRÁCTICA

1. Repetir el ejercicio anterior pero considerando que la curva que une los puntos $\hat{P}_1$ y $\hat{P}_2$ es una poligonal como se muestra en la siguiente figura de n puntos, siendo n un número arbitrario. FIGURA ILUSTRATIVA

EJERCICIOS PROPUESTOS

1. entrada 4