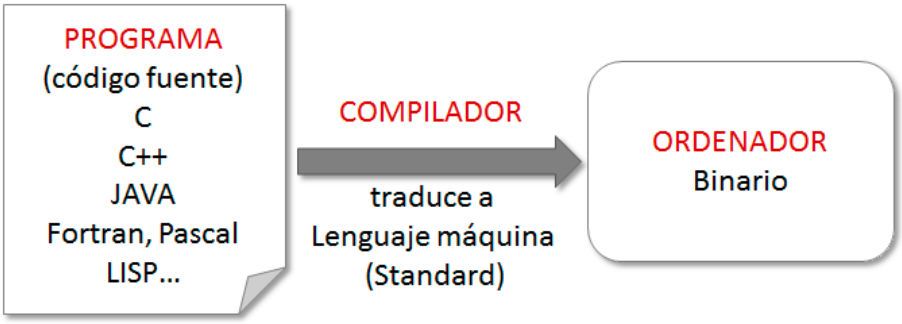
Universidad de Oviedo
Dpto. Matemáticas

PRÁCTICA 1

1. Introducción a la programación en MATLAB

Por lo general, el proceso de programación sigue el siguiente esquema:

- Fase 1: El compilador traduce el código fuente del Programa a Leanguaje máquina (standard). El ordenador procesa en código binario.



- Fase 2: LINK o relación entre objetos - Ejecutable (software).

MATLAB no sigue este esquema ya que es directamente un ejecutable (un programam precompilado). Los ficheros .m son a la vez código fuente y ejecutables. Existen dos tipos de ficheros .m

- Scripts: no tienen argumentos de entrada (Inputs) ni de salida (Outputs). Inputs, Outputs directamente en el Workspace.
- Functions: rutinas y subrutinas. Tienen argumentos de entrada y de salida. Si no proporcionan Outputs se denominan "voids".

En la programación en general, encontramos varios componentes:

- Variables: Arrays (vectores), Matrix (matrices), Structures, cell-arrays

- Estructuras iterativas y de control: for (bucle), if (condicional), while, swich-case.
- Funciones y Scripts
- Inputs/Outputs

Ejemplo de Structure:

```
alumno.name = 'Juan';
alumno.DNI = '5026623C';
alumno.notas = [7.0, 5.6, 9.5];
```

Escribir los comandos directamente en la ventana de trabajo de MATLAB resulta cómodo para ejecutar unas pocas órdenes. Sin embargo, cuando tenemos que ejecutar un gran número de instrucciones de forma consecutiva, necesitamos crear un **programa**.

En MATLAB, un programa es un fichero con la extensión `.m` y en el cual hay escritos comandos que MATLAB pueda entender. Para crear un programa hay que usar un **editor** de texto. MATLAB trae uno incorporado que es especialmente útil, ya que resalta en diferentes colores comandos, variables, mensajes, ... Para activarlo se puede pinchar con el ratón en el icono de la barra de herramientas. Se abre el editor de textos de MATLAB. Escribimos:

```
A=[3,2;1,0];
% Esto es una matriz.
b=[7;3];
% Podemos poner comentarios empezando la frase con el símbolo %
x=A\b
```

Lo guardamos en nuestro directorio, por ejemplo una carpeta *Master*, con un nombre válido: `prueba1.m`. Para ejecutarlo volvemos a la ventana de trabajo y seleccionamos en la barra de arriba la carpeta donde está el fichero. Tecleamos el nombre del programa, sin extensión,

```
» prueba1
```

y MATLAB ejecuta las tres órdenes de manera consecutiva (se salta los comentarios) y nos devuelve sólo el valor de `x`.

1.1. Entrada y salida de datos de un programa

Si quisieramos cambiar `A` o `b` en el programa anterior tendríamos que modificarlo. Normalmente, un usuario no modifica un programa, sino que le puede enviar y pedir datos de alguna manera. En MATLAB hay dos maneras principalmente de realizar esta tarea:

1. Por pantalla, mediante las órdenes `input` y `disp`.
2. Mediante el uso de funciones.

Una función es un programa especial que pide unos datos y devuelve otros. El funcionamiento básico es como el de una función en matemáticas. Por ejemplo, la función seno. Si le metes como dato $\pi/2$, la respuesta es 1.

Programando ficheros `.m` es frecuente que queramos presentar algún mensaje o dato en la pantalla, e incluso que el propio programa nos pida parte de los datos por pantalla. Esto puede hacerse con las órdenes siguientes:

Función	**Salida**
`input`	Permite introducir un dato desde el teclado La ejecución del programa continúa al pulsar ↩
`input('Cadena de caracteres')`	Realiza la misma operación que la instrucción anterior, pero mostrando en pantalla el mensaje *Cadena de caracteres*, a la espera de recibir datos
`disp('Cadena de caracteres')`	Muestra en pantalla la `Cadena de caracteres`. No espera datos.
`disp(x)`	Muestra en pantalla el valor de la variable x
`fopen(filename)`	Abre fichero
`fread (fileID)`	Lee fichero
`fprint(filename)`	Escribe en el fichero
`print(filename,formattype)`	Últimas versiones
`fclose(fileID)`	Cierra fichero

2. Estructuras de control

En los programas de MATLAB se manejan principalmente tres estructuras de control:

- Decisión: `if ... elseif ... else ... end`
- Repetición un número fijo de veces: `for ... end`
- Repetición bajo condiciones: `while ... end`

Ejemplo 1 *Programar una función que calcule mediante un* `for` *el factorial de un número.*

```
function f=facto(n)
% Función que calcula el factorial de un número
if n~=round(n)|n<0
      error('Debe introducir un entero no negativo')
end
f=1;
for k=1:n
      f=f*k;
end
```

El uso de `for` es cómodo, pero ralentiza la ejecución de los programas. En MATLAB casi siempre hay alternativas más rápidas. Vamos a comparar nuestro programa con la orden `prod`, que multiplica todos los elementos de un vector, mediante el uso de `tic ... toc`

```
» tic, facto(20);toc
» tic, prod(1:20);toc
```

Ejemplo 2 *Resolver la ecuación de* 2^o *grado.*

Diagrama de flujo: $coef \rightarrow \Delta = b^2 - 4ac \begin{cases} \Delta > 0, & \text{2 raices reales distintas;} \\ \Delta = 0, & \text{raiz única;} \\ \Delta < 0, & \text{raices complejas.} \end{cases}$

```
function [ x ] = second_order(coef)
% Resolver la ecuación de 2° grado ax^2+bx+c=0
% Inputs: coef=[a,b,c] los coeficientes de la ecuación
% Outputs: x=[x1,x2]
x=zeros(2,1);
a=coef(1); b=coef(2); c=coef(3);
delta=b^2-4*a*c;
x(1)=(-b+sqrt(delta))/(2*a);
x(2)=(-b-sqrt(delta))/(2*a);

if delta >0
    disp('existen 2 raices reales diferentes')
elseif delta <0
    disp('existen 2 raices complejas')
else
    disp('una única raiz real')
end
```

Ejemplo 3 *Dada una matriz A, calcular la suma de los valores absolutos de sus vectores columna.*

Este problema se podría resolver directamente mediante el comando:

```
sum(abs(A))
```

MATLAB actúa sobre las columnas de A por defecto, a menos que se trate de un vector cuando la orden sum devuelve la suma de sus elementos:

```
v=[0 -2 1 3 4];
>> sum(abs(v))

ans =

    10
```

Otra solución para este problema sería crear nuestra propia función:

```
function [ s ] = suma_col( A )
% suma de los valores absolutos de los vectores columna de A
[m,n]=size(A);
% if (m==1)|(n==1) % El operador lógico OR |
if or(isequal(m,1),isequal(n,1))
    disp('Input is incorrect')
    return
end
```

```
s=zeros(1,n);
for j=1:n
    for i=1:m
        s(j)=s(j)+abs(A(i,j));
    end
end
```

Ejercicio 1 *Escribe, usando la estructura* `for ... end` *una función* `ejercicio1.m` *de una variable de entrada* `n` *(número natural* impar*), que devuelva la suma de todos los cuadrados de los números impares desde el uno hasta el* `n`*. En el caso de que el usurio metiera un número no natural impar, dar un error explicativo.*

3. Proyección ortogonal. Mínimos cuadrados

Función	Salida
`int(f,x,a,b)`	Devuelve la integral definida de la función `f` con respecto a `x` entre `a` y `b`.
`plot(x,y,S)`	Dibuja el vector `y` frente al vector `x` en un determinado color y un determinado tipo de línea indicado en S. Por ejemplo, plot(x,y,'c+') dibuja una línea azul de +, que constituye la gráfica de `y` frente a `x`.
`hold on`	Guarda el dibujo actual y todas las propiedades de sus ejes, de modo que las órdenes de dibujo de la gráfica siguiente se añadan a la que ya existe.

3.1. Proyección ortogonal

Ejemplo 4 *Aproximar la función* $f(x) = sen(\pi x)$ *en el intervalo* $[-1, 1]$ *mediante polinomios de* $\mathbb{R}_3[x]$ *tomando como producto escalar*

$$f \cdot g = \int_{-1}^{1} f(x)\, g(x)\, dx$$

Para aproximar f se busca un polinomio $P \in \mathbb{R}_3[x]$ que minimice la distancia $\|sen(\pi x) - P\|$, lo que equivale a decir que $sen(\pi x) - P$ es ortogonal a $\mathbb{R}_3[x]$, para lo cual basta que $sen(\pi x) - P$ sea ortogonal a una base de $\mathbb{R}_3[x]$, por ejemplo a la canónica

$$\begin{cases} (sen(\pi x) - P) \cdot 1 = 0 \\ (sen(\pi x) - P) \cdot x = 0 \\ (sen(\pi x) - P) \cdot x^2 = 0 \\ (sen(\pi x) - P) \cdot x^3 = 0 \end{cases} \leftrightarrow \begin{cases} sen(\pi x) \cdot 1 = P \cdot 1 \\ sen(\pi x) \cdot x = P \cdot x \\ sen(\pi x) \cdot x^2 = P \cdot x^2 \\ sen(\pi x) \cdot x^3 = P \cdot x^3 \end{cases}$$

.

La aproximación buscada es la proyección ortogonal de f sobre el espacio $\mathbb{R}_3[x]$. Como $P \in \mathbb{R}_3[x]$, se puede expresar como $P = a_0 1 + a_1 x + a_2 x^2 + a_3 x^3$, y:

$$\begin{cases} \left(a_0 1 + a_1 x + a_2 x^2 + a_3 x^3\right) \cdot 1 = sen(\pi x) \cdot 1 \\ \left(a_0 1 + a_1 x + a_2 x^2 + a_3 x^3\right) \cdot x = sen(\pi x) \cdot x \\ \left(a_0 1 + a_1 x + a_2 x^2 + a_3 x^3\right) \cdot x^2 = sen(\pi x) \cdot x^2 \\ \left(a_0 1 + a_1 x + a_2 x^2 + a_3 x^3\right) \cdot x^3 = sen(\pi x) \cdot x^3 \end{cases}$$

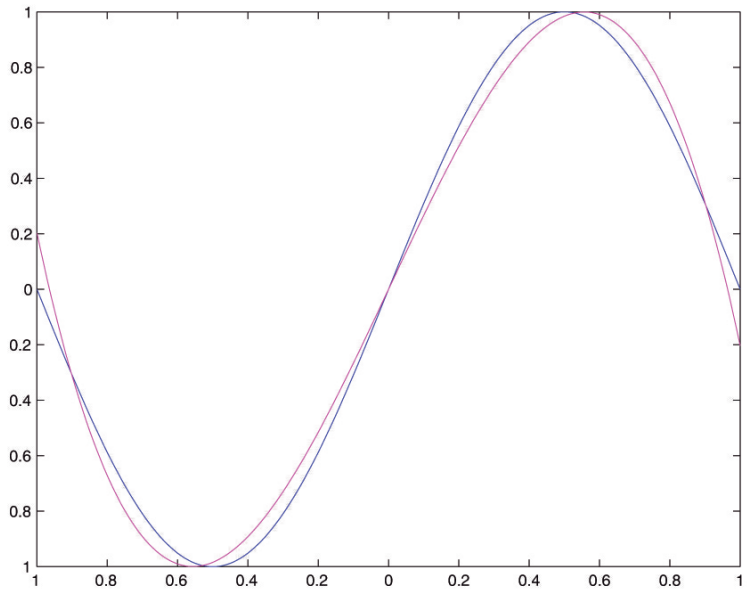
o bien

$$\begin{pmatrix} 1\cdot 1 & 1\cdot x & 1\cdot x^2 & 1\cdot x^3 \\ 1\cdot x & x\cdot x & x\cdot x^2 & x\cdot x^3 \\ 1\cdot x^2 & x\cdot x^2 & x^2\cdot x^2 & x^2\cdot x^3 \\ 1\cdot x^3 & x\cdot x^3 & x^2\cdot x^3 & x^3\cdot x^3 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} sen(\pi x)\cdot 1 \\ sen(\pi x)\cdot x \\ sen(\pi x)\cdot x^2 \\ sen(\pi x)\cdot x^3 \end{pmatrix}$$

```
» syms x
» v=[1 x x^2 x^3]
» H=v.'*v  % Matriz simbólica del producto escalar
» G=int(H,x,-1,1) %Matriz de Gramm
» D=v.'*sin(pi*x) % Matriz simbólica segundo miembro
» b=int(D,x,-1 ,1) % Matriz numérica segundo miembro
» a=inv(G'*G)*G'*b;% Solución del sistema (Los coeficientes del polinomio buscado)
```

Representación gráfica de f y de su aproximación $f_{aprox} = P$:

```
» P=a'*v' % El pollinomio que aproxima la función
» xnum=-1:0.01:1; % generamos valores en el intervalo [-1,1] para dibujar la función y el polinomio
» y=sin(pi*xnum);
» P=subs(P,x,xnum)
» plot(xnum,y,'b',xnum,P,'m')
```



3.2. Ajuste de datos por mínimos cuadrados

En los problemas basados en datos reales se plantea el sistema lineal $Ax = b$, donde las columnas de A (el espacio $Col(A)$) contienen los datos resultados de unas mediciones. Se trata de problemas puramente sobredeterminados ($m \gg n$).

De forma general, el vector de valores $\vec{b} \notin Col(A)$ y en consecuencia este tipo de problemas no tiene solución, por lo que se busca una solución aproximada $\vec{x}_{LS}$ que minimice el error $E = \|Ax - b\|$ (que representa la distancia entre b y Ax).

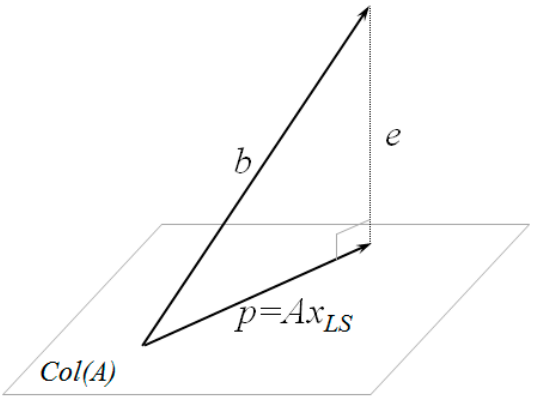
La idea es hallar $\vec{x}_{LS}$ por mínimos cuadrados (Least Squares) y predecir $A\vec{x}_{LS}$, cometiendo un error $\vec{e} = \vec{b} - A\vec{x}_{LS}$. Esta solución se halla proyectando ortogonalmente $\vec{b}$ sobre el espacio $Col(A)$. La proyección ortogonal de $\vec{b}$ sobre este subespacio es $\vec{p} = A\vec{x}_{LS}$.

Tenemos $\vec{b} = A\vec{x}_{LS} + \vec{e}$, con $\vec{e} \perp Col(A) \Rightarrow A^T\vec{e} = 0 \Rightarrow \vec{e} \in Ker(A^T)$. Pero $\vec{e} = \vec{b} - A\vec{x}_{LS}$, de donde $A^T(\vec{b} - A\vec{x}_{LS}) = 0 \Rightarrow A^TA\vec{x}_{LS} = A^T\vec{b}$. Como A^TA es invertible cuando las columnas de A son linealmente independientes (lo que ocurre en estos casos), obtenemos la *mejor estimación*:

$$\boxed{\vec{x}_{LS} = (A^TA)^{-1}A^T\vec{b}}$$

y la proyección ortogonal sobre $Col(A)$ es:

$$\boxed{A\vec{x}_{LS} = A(A^TA)^{-1}A^T\vec{b}}$$



Ejercicio 2 *En la fabricacion de un cierto producto X, la cantidad de un cierto compuesto β presente es controlada por la cantidad del ingrediente α utilizado en el proceso. Al fabricar un kilogramo de X se registran la cantidad α utilizada y la cantidad β presente. Se obtienen los siguientes datos*

Cantidad α utilizada	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
Cantidad β presente	*4.5*	*5.5*	*5.7*	*6.6*	*7*	*7.7*	*8.5*	*8.7*	*9.5*	*9.7*

Suponiendo que la relacion entre la cantidad α y la cantidad β esta dada por una ecuacion lineal $\beta = a\alpha + b$, determinar dicha recta utilizando el metodo de los minimos cuadrados. Utilizar la ecuacion obtenida para predecir la cantidad β presente en un kilogramo de X si se utilizan 30 unidades de α por kilogramo de X.

Al tratar de ajustar los datos que tenemos mediante la recta $\beta = a\alpha + b$, se obtiene el siguiente sistema lineal:

$$\begin{cases} a\alpha_1 + b = \beta_1 \\ \dots \\ a\alpha_n + b = \beta_n \end{cases} \Leftrightarrow \begin{pmatrix} \alpha_1 & 1 \\ \dots & \\ \alpha_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \dots \\ \beta_n \end{pmatrix}$$

Los parámetros a y b se calculan dando una solución por mínimos cuadrados del sistema lineal, y la cantidad de producto β presente en un kilogramo de X si se utilizan 30 unidades de α por kilogramo de X es $\beta_{pred} = a \cdot 30 + b$.

Ejercicio 3 *La tabla siguiente muestra la cantidad de un cierto contaminante, y, con respecto a la cantidad normal encontrada en un punto geografico, para una cierta cantidad de aire y en intervalos de media hora:*

t	1	1.5	2	2.5	3	3.5	4	4.5	5
y	-0.15	0.24	0.68	1.04	1.21	1.15	0.86	0.41	-0.08

La grafica de las medidas sugiere una relacion de tipo polinomico. Encontrar el polinomio de grado 2 ($y = at^2 + bt + c$) que produzca un buen modelo a partir de estos datos.

Ejercicio 4 *Se lanza un cuerpo desde una posición inicial $x_0 = -50$ con una velocidad inicial $v_0 = 100m/s$ que describirá una trayectoria parabólica hasta su caida al suelo.*

1. *Generar un conjunto de datos para los primeros $15s$, calculando el espacio recorrido x a cada $0{,}1s$. Generar un segundo conjunto de datos introduciendo un ruido*
2. *Ajustar los datos por mínimos cuadrados para predecir los parámetros x_{0p}, v_{0p} y g_p para los dos conjuntos de datos generados. Comparar los resultados.*
3. *Predecir la distancia donde el cuerpo llega al suelo.*

```
% Trayectoria parabólica
% Generar datos
s0=-50;v0=100;g=9.8; % datos iniciales
t=[1:0.1:15]';
s=s0+v0*t-1/2*g*t.^2;
plot(t,s), hold on
% Añadir ruido
noise=0.03;
mu=2;
sr=s+noise*mean(s)*rand(length(t),1)-mu;
plot(t,sr,'m-o'), hold on
% Ajuste por Mínimos Cuadrados
A=[ones(length(t),1) t -1/2*t.^2];
b=sr;
xLS=inv(A'*A)*A'*b;
s0_pred=xLS(1)
v0_pred=xLS(2)
g_pred=xLS(3)
sr_pred=A*xLS;
plot(t,sr_pred), hold on
title('Trayectoria parabólica')
xlabel('t')
ylabel('s')
legend('datos', 'datos ruido', 'datos pred')
% Predicción de la caída
tm=t(end);sm=sr(end);
while sm>0
    tm=tm+0.1;
    sm=s0_pred+v0_pred*tm-1/2*g_pred*tm^2;
    plot(tm,sm,'m.')
    pause(0.1)
end
```

Ejercicio 5 *Realizar una modelización por mínimos cuadrados de los siguientes datos de las ventas del iPad, mediante el modelo:* $y^*(t) = a_0 + a_1 t + a_2 log_{10}(t)$. *Predecir las ventas del año 2016 y calcular el intervalo de confianza.*

Año	*2007*	*2008*	*2009*	*2011*	*2012*	*2015*
Ventas (euro)	*40010*	*45205*	*49823*	*54102*	*54200*	*54900*