7. Funciones

- Cuando escribimos un programa/script en matlab desde allí podemos llamar a funciones que matlab ya tiene implementadas (Ej. *sin*, *mean*, *sort*,...) o a funciones que nosotros hemos programado
- El programa/script y la función se escriben en ficheros **.m**
- Las funciones aceptan argumentos de entrada **in** y producen unos resultados de salida **out**
- 1ª línea del archivo **.m** que contiene la función (Ej. **mifuncion.m**):
  - **function** **out=mifuncion(in)**
  - **function** **[out1,....,outN]=mifuncion(in1,...,inM)**
- Para llamar a la función desde el programa/script (Ej. **principal.m**):
  - **var=mifuncion(dato)**
  - **[var1,....,varN]=mifuncion(dato1,...,datoM)**

  Los parámetros de entrada y salida pueden tener **nombres diferentes** en la función y en el programa/script que la llama
- Las variables definidas en **mifuncion.m** son **locales**. Pueden ser **globales** si la definimos: `global nombre_var`

7. Funciones

Ej.1 Calcular precio de un producto que se incrementa un porcentaje

- Programa/script principal o línea de comandos:

```
>> precio=1000; inc=16/100;
>> preciofin=pvt(precio,inc);
>> preciofin
```

ans=1160

- Función **pvt.m**:

```
function precio_final=pvt(precio_inicial, incremento);
precio_final=precio_inicial*(1+incremento);
```

7. Funciones

Ej.2 Obtener las raíces de un polinomio cuadrático: $ax^2 + b * x + c = 0$

- Programa/script principal o línea de comandos:

```
>> [x1,x2]=raices(1,3,2);
>> [x1,x2]
```

ans: x1=-1, x2=-2

- Función **raices.m**:

```
function [x1,x2]=raices(a,b,c);
d=b^2-4*a*c;
if a~=0
  x1=(-b+sqrt(d))/(2*a);
  x2=(-b-sqrt(d))/(2*a);
elseif b~=0
  x1=-c/b; x2='no hay'
  disp('Ec lineal, solo una raiz');
else c~=0
  disp('Ec imposible');
end
```

7. Funciones

- A veces, necesitaremos poner una **función como parámetro de entrada de otra función**. Indicaremos que es una función poniendo delante el símbolo **@**, que se llama **function handle**

Ej.3 Resolver integral mediante la función de matlab quad:

$$I = \int_0^{\pi} \sin(x)dx$$

```
>> I=quad(@sin,0,pi)
```

O también:

```
>> integrando=@sin;
>> I=quad(integrando,0,pi) ans=2
```

7. Funciones

Ej.4 Calcular numéricamente la derivada de una función en un punto

```
>>resultado=derivada(@mifuncion_inicial,3)
```

ans=6.00000100092757

derivada.m

```
function der=derivada(nomfun,x0)
dx=1e-6;
f2=nomfun(x0+dx);
f1=nomfun(x0);
der=(f2-f1)/dx;
```

mifuncion_inicial.m

```
function fun=mifuncion_inicial(x)
fun=x^2;
```

7. Funciones

- También se puede evaluar una función usando **feval**:
  - `[y1,y2,....yN]=feval(F,x1,x2,....xN);`
  - **F** puede ser una **variable alfanumérica/caracter** o un **function handle @**

Ej.5 Calcular x^2 en $x = 3$

```
>>f1=feval('mifuncion_inicial',3)   ans=9
>>f1=feval(@mifuncion_inicial,3)   ans=9
```

Ej.6 Calcular numéricamente la derivada de una función en un punto

```
>>resultado=derivada('mifuncion_inicial',3)
```

ans=6.00000100092757

derivada.m

```
function der=derivada(nomfun,x0)
dx=1e-6;
f2=feval(nomfun,x0+dx);  f1=feval(nomfun,x0);
der=(f2-f1)/dx;
```

8. Input/Output

ENTRADAS

load. Leer datos de un fichero ASCII: `>> load nombre_fichero.dat;`
Se puede asignar a una varible: `>> y=load nombre_fichero.dat;`

input. Pedir por pantalla variable `>> x = input('Introduce x');`

SALIDAS

disp. Mostrar por pantalla:
`>> disp('una parrafada'); >> disp(x)`

num2str. Transformar variable numérica en caracter.

```
>> x=0.35; x_car=num2str(x)
>> disp(['El resultado es ' x_car]);
```

8. Input/Output

fprintf. Escribir con formato en pantalla.

```
>> fprintf(formato_lista, lista_salida);
```

Formatea las variables de lista_salida con formato especificado en formato_lista.

```
>> fibo=5;
>> fprintf('Primer termino de sucesion es %i.  \n', fibo);

>> x=0:0.1:1; y=exp(x); fprintf('%5.2f %5.2e \n',[x;y])
```

fopen/fclose. Escribir en fichero.

```
>> x=0:0.1:1; z=[x; exp(x)];
>> fid=fopen('practica1.txt','w');
>> fprintf(fid,'%5.2f %5.2e \n',z)
>> fclose(fid);
```