OBJETIVOS DE APRENDIZAJE

- **Estructuras iterativas: Bucles *for***
  - **for <loopvar> = <range> ... end**
  - **Estructuras iterativas/condicionales anidadas**
  - **while**

EJERCICIO 7.1 *FOR*

Usa una sentencia *for* para imprimir en pantalla todos los múltiplos de 5 existentes desde el 0 al 25 junto con su raíz cuadrada (usando 4 decimales).

EJERCICIO 7.2 *FOR* / IF

Ejecutar una sentencia for que visualice los números desde el -5 hasta el +5 en incrementos de 1, junto con su raíz cuadrada si es positivo o con su raíz cúbica si es negativo. Cada valor numérico será mostrado con el formato adecuado.

EJERCICIO 7.3 *FOR ANIDADOS CON IF*

Crea un script que genere una matriz 5 × 5 matriz tal que:

1. Todos los elementos en las columnas y filas pares tendrán el valor 1.
2. El resto tendrán el valor 0

$$\begin{matrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{matrix}$$

EJERCICIO 7.4 REPRESENTAR GRÁFICAMENTE (PLOT) UNA SEÑAL

En este ejercicio, queremos representar gráficamente una función en el dominio *[0 , 2*pi]* con diferentes resoluciones (número of puntos diferente).

1. *En primer lugar, crearemos una function denominada discretizeSignal que, dados un número de puntos (numPoints), devuelve 2 vectores: x e y. El primer vector x contendrá numPoints elementos de 0 to 2*pi, mientras que el segundo vector y tendrá* $y=\cos(x)*\sin(x)^2$. El prototipo de la función es la siguiente:

```
function [x,y] = discretizeSignal(numPoints)
```

Respecto al control de errores, la función comprobará que *numPoints* sea positivo y mayor que 1. En caso contrario, *x* e *y* no tendrán ningún valor asignado (serán vectores vacíos).

Nota: cuando sea posible, es mejor usar operaciones vectoriales en lugar de una estructura de tipo for.

2. Programaremos un script que llame a *discretizeSignal* con los siguientes números de puntos: 3, 5, 10 y 20. Después de llamar a la función, representaremos gráficamente (plotear) los vectores devueltos. Todos los plots deberían estar en el mismo plot (Pista: usar *hold on* y *hold off*).

EJERCICIO 7.5 BUSCAR UN ELEMENTO

Escribe un script que defina el siguiente vector:

```
vector = [ 2 5 2 49 -3 59 3 9 ]
```

A continuación, pregunta a la persona usuaria por un elemento. El script imprimirá la posición del elemento en el vector. Si no se encuentra en el vector, deberá imprimir un mensaje al respecto.

Ejemplo de ejecución #1:

```
>> searchForElement
>> What element do you want to search for? 59
>> Element 59 was first found in position 6
```

Ejemplo de ejecución #2:

```
>> searchForElement
>> What element do you want to search for? 59
>> Element 59 was not found in the vector
```

EJERCICIO 7.6 SCRIPT MENÚ

Implementa un script que muestre el siguiente menú:

```
>>MENU##############
>>1. Exercise 7.1
>>2. Exercise 7.2
>>3. Exercise 7.3
>>4. Exercise 7.4
>>5. Exercise 7.5
>>6. Exit
>>##################
>>What exercise do you want to run?
```

Tras mostrar en pantalla el menú, la persona usuaria introducirá una de las opciones y, dependiendo de la opción seleccionada, se ejecutará uno de los ejercicios (anteriormente implementados).
Este proceso será repetido (mostrar el menú y procesar la entrada como input) hasta que la última opción sea seleccionada. Antes de mostrar el menú, la ventana de comandos será reseteada (clear).