5. Representación gráfica

- **PLOT**:
  `plot(x,y,'color_style_marker','PropertyName',PropertyValue)`

Ej.1
```
>> t=(0:pi/100:2*pi);
>> y=sin(t);
>> plot(t,y)
```

Ej.2
```
>> x=pi*(-1:0.01:1);
>> y=x.*sin(x);
>> plot(x,y)
```

Ej.3
```
>> figure(1);
>> t=(0:pi/100:2*pi);
>> y1=sin(t);y2=sin(t-.25);
    y3=sin(t-.5);
>> plot(t,y1)
>> hold on
>> plot(t,y2)
>> hold off
>> figure(2);
>> plot(t,y3)
```

color_style_marker:
Cadena de caracteres de longitud 1, 2 o 3 formada por color, tipo de línea y tipo de punto.

Color:
`'m','y','r','g','b','w','k'`

Linestyle:
`'-','--',':','-.','none'`

Marker:
`'+','o','*','x'`

PropertyName:
'LineWidth'
'MarkerEdgeColor'
'MarkerFaceColor'
'MarkerSize' ...

5. Representación gráfica

Ej.4

```
>> x = -pi:pi/10:pi;
>> y = tan(sin(x)) - sin(tan(x));
>> plot(x,y,'--rs','LineWidth',2, 'MarkerEdgeColor',
'k', 'MarkerFaceColor','g', 'MarkerSize',10)
```

- **AXIS**: `axis([XMIN XMAX YMIN YMAX])`, `axis auto`, `axis on/off`, `axis equal`
- **title**: `title('Titulo grafica')`
- **xlabel**: `xlabel('nom_var_x')`
- **ylabel**: `ylabel('nom_var_y')`
- **grid** `on`, **grid** `off`
- **loglog**`(x,y)`, **semilogy**`(x,y)`, **semilogx**`(x,y)`
- **subplot**`(n m p)`
- **text**`(x, y, 'string' )`

5. Representación gráfica

- **Imprimir gráficas:** `print [-opciones] ficherosalida`

```
>>print -deps nombrefigura.eps
>>print -depsc nombrefigura.eps
```

- **FPLOT**: `fplot('funcion',[xmin xmax ymin ymax])`

Ej.5

```
>> fplot('sin(x)',[0 2*pi]);
>> hold on
>> fplot('cos(x)',[0 2*pi]);
>> hold off
```

`hold on` → Mantiene en la ventana gráfica los dibujos anteriores

`hold off` → Olvida dibujos anteriores y dibuja en ventana nueva

6. Control de flujo

- **Control de flujo** $\rightarrow$ Proceso de toma de decisiones dentro de un programa

  Condiciones $\rightarrow$ Preguntas básicas a las que se puede responder sí o no. Para implementar control de flujo se usan expresiones que permiten comparar dos variables entre si o con un valor fijo.
  Las condiciones se construyen con **operadores relacionales**:
  $>$ $<$ $==$ $\sim=$ $>=$ $<=$
  Se pueden combinar con **operadores lógicos**:
  $\&$ $\rightarrow$ i $\quad$ $|$ $\rightarrow$ o $\quad$ $\sim$ $\rightarrow$ no

  Ramificaciones $\rightarrow$ Dependiendo de la condición el programa decide lo que hará. Si la condición es **verdadera** se ejecutan determinadas sentencias. El **diagrama de flujo se ramifica.**

6. Control de flujo

- Bucles WHILE

Permiten repetir una serie de instrucciones hasta que deja de cumplirse la condición lógica

```
while condicion
   instrucciones;
end
```

- Bucles FOR

Permiten repetir una serie de instrucciones un número fijo de veces

```
for i=n1:n2:n3
   instrucciones;
end
```

- Sentencias IF

Permiten saltar de una sentencia a otra si se cumple una condición que puede depender de los valores que genera el programa

```
if condicion
   instrucciones;
elseif condicion
   instrucciones;
....
else
   instrucciones;
end
```

- Sentencia BREAK

Se puede salir de un bucle FOR o sentencia IF

6. Control de flujo

Ej.1 Calcula la suma $\sum_{n=1}^{10} \frac{1}{n}$

Manera 1

```
>> s=0;
>> for n=1:10
>>    s=s+1/n;
>> end
```

Manera 2 Más compacto!

```
>> s=sum(1./[1:10])
```

Ej.2 Calcular la expresión $\sqrt{y-x}/x$ mientras sea real ($x \neq 0$ y $y > x$).

```
>> x=1; y=6;
>> while (x~=0) & (x<y)
>>    b=sqrt(y-x)/x;
>>    x=x+1;
>> end
```