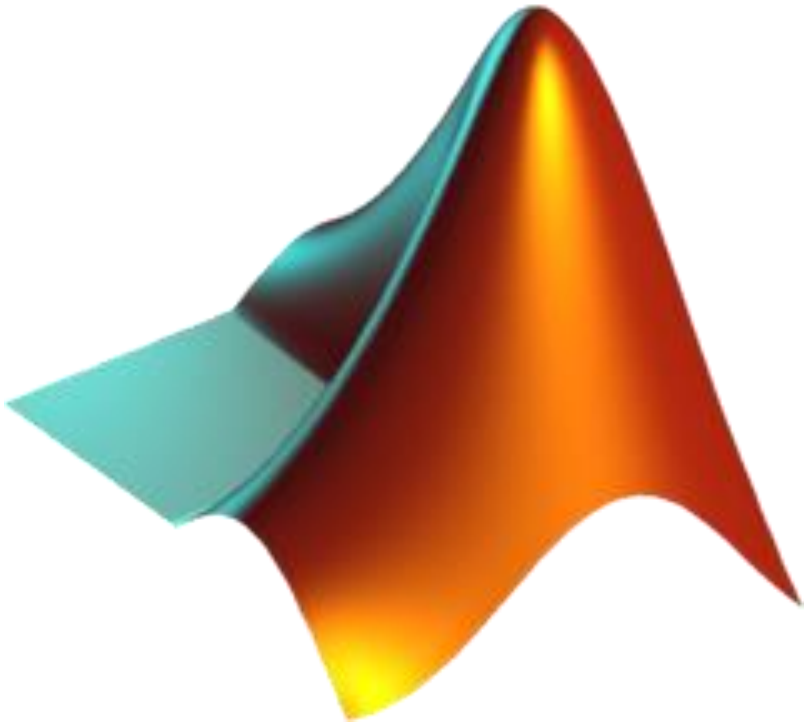
Introducción a MATLAB & SIMULINK



Introducción

- ¿Qué es Matlab?, MATrix LABoratory

- Es un lenguaje de programación (inicialmente escrito en C) para realizar cálculos numéricos con **vectores y matrices**. Como caso particular puede también trabajar con números escalares, tanto reales como complejos.

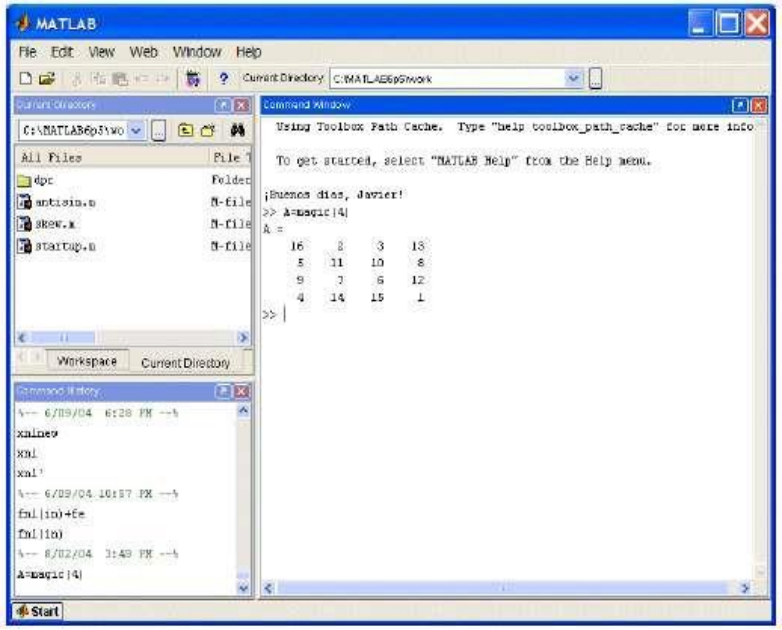
- Cuenta con paquetes de funciones especializadas en el calculo matricial.

- **Entorno de programación caracterizado por**:
  - lenguaje interpretado (fácil de usar, pero menos eficiente que los compilados).
  - No fuertemente tipado.
  - Muy buenas características para representación gráfica.
  - Útil para prototipado de programas.

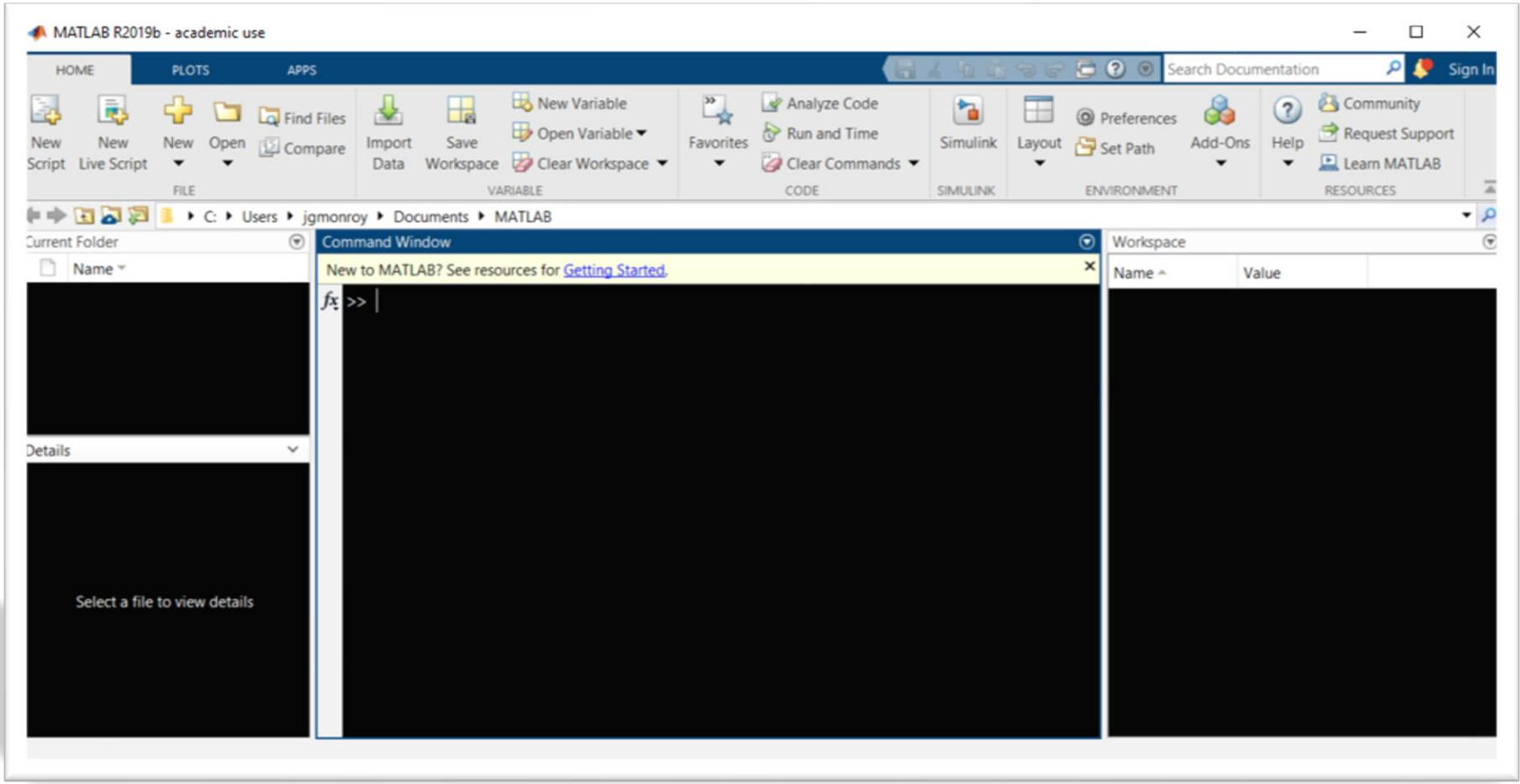
Introducción

Elementos básicos del escritorio de Matlab

- **Command Window**: Donde se ejecutan todas las instrucciones y programas. Puede usarse para cálculos rápidos.
- **Command History**: Muestra los últimos comandos ejecutados en el Command Window. Muy útil para comprobar las acciones recientes y re-ejecutar instrucciones.
- **Workspace**: Muestra las variables que se están usando, su tipo, sus dimensiones (si son matrices) y su valor actual.
- **Current directory**: Ventana de navegador que muestra el path actual y los ficheros en el mismo.



Matlab 2019


MATLAB R2019b - academic use
HOME
PLOTS
APPS
Search Documentation
Sign In
New Script
New Live Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Favorites
Analyze Code
Run and Time
Clear Commands
Simulink
Layout
Preferences
Set Path
Add-Ons
Help
Community
Request Support
Learn MATLAB
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
C: ▸ Users ▸ jgmonroy ▸ Documents ▸ MATLAB
Current Folder
Name
Details
Select a file to view details
Command Window
New to MATLAB? See resources for Getting Started.
>>
Workspace
Name
Value

Tipos de Datos - Numéricos

- **No fuertemente tipado** → no hace falta definir el tipo de variable
  - Números enteros: a=2
  - Números reales: x=-35.2
    - Máximo de 19 cifras significativas
    - 2.23e-3 = $2.23*10^{-3}$
  - Números complejos: a = 2.3 + 8j (cuidado de no usar "i" o "j" como variables)
  - Case-Sensitive: se distingue entre mayúsculas y minúsculas: x=5, X=7
  - Para eliminar alguna variable se ejecuta>> clear variable1 variable2
  - Si se quieren borrar todas las variables: >> clear
  - Constantes características: pi= π, NaN (not a number, 0/0), Inf=∞

Tipos de Datos - Operaciones

Operaciones Aritméticas Elementales:

- Suma: +, Resta -
- Multiplicación: *, División: /
- Potencias: ^ (acento circunflejo)
- **Orden de prioridad**: Potencias, divisiones y multiplicaciones y por ultimo sumas y restas. Usar () para cambiar la prioridad.

Otras Operaciones Aritméticas:

- exp(x), log(x), log2(x) (en base 2), log10(x) (en base 10), sqrt(x)
- Funciones trigonométricas: sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x) (entre –pi y pi)
- abs(x) (valor absoluto), int8(x), int16(x)…int64(x) (parte entera), round(x) (redondea al entero más próximo), sign(x) (función signo)
- Funciones para números complejos: real(z) (parte real), imag(z) (parte imaginaria), abs(z) (módulo), angle(z) (ángulo), conj(z) (conjugado)

Tipos de Datos - Matrices

Definición de Vectores:

- **Vectores fila**; elementos separados por espacios en blancos o comas y delimitados por corchetes []

  >> v =[2 3 4]

- **Vectores columna**: elementos separados por **punto y coma** (;)

  >> w =[2;3;4;7;9;8]

- Dimensión de un vector w: **length(w)**

- **Generación de vectores fila**:
  - Especificando el incremento **h** de sus componentes **v=a:h:b**
  - Especificando su dimensión **n: linspace(a,b,n)** (por defecto n=100)
  - Componentes logarítmicamente espaciadas **logspace(a,b,n)** (n puntos logarítmicamente espaciados entre 10^a y 10^b. Por defecto n=50)

Tipos de Datos - Matrices

Definición de Matrices:

- Tienen tamaño dinámico. No hace falta establecer de antemano su tamaño.

- Los elementos de una misma fila están separados por blancos o comas.
  Las filas están separadas por punto y coma (;).

```
» M=[3 4 5; 6 7 8; 1 -1 0]
```

```
M=  3   4   5
    6   7   8
    1  -1   0
```

- **Matriz vacía**: M=[ ];
- **Matriz de ceros:** zeros(n,m)
- **Matriz de unos:** ones(n,m)
- **Matriz identidad:** eye(n,m)
- **Matriz de elementos aleatorios:** rand(n,m)

IMPORTANTE!!
Creación de Matrices.

- **Añadir matrices**: [X Y] columnas, [X; Y] filas
- **Eliminar una columna**: M(:,1)=[ ], **una fila**: M(2,:)=[ ];

Tipos de Datos - Matrices

Acceso a Elementos:

- **Los índices comienzan en 1** → no existe fila 0 ni columna 0.
- Para el acceso se emplean paréntesis ()
  - M(1,3) → Acceso a un elemento (fila, columna)
  - M(1,2:3) → Submatriz (fila 1, columnas 2 hasta 3)
  - M(1:2,2:3) → Submatriz (fila 1 a 2, columnas 2 a 3)
  - M(2,:) → Fila 2 completa
  - M(:,3) → Columna 3 completa
  - M(1,3:-1:1) → Se pueden definer rangos con paso X
  - M(3,1:end) → Submatriz (fila 3, columnas de la 1 a la última)
  - M(3,1:end-1) → Submatriz (fila 3, columnas de la 1 a la penúltima)
- Cambiar el valor de algún elemento:
  - M(2,3)=1;
  - M(2,:)=[5 5 5];

Operaciones con Matrices

- + adición o suma
- – sustracción o resta
- * multiplicación matricial
- .* producto elemento a elemento
- ^ potenciación
- .^ elevar a una potencia elemento a elemento
- \ división-izquierda
- / división-derecha
- ./ y .\ división elemento a elemento
- ' matriz traspuesta: B=A'
- det() determinante de la matriz cuadrada
- size() tamaño de la matriz en (filas, columnas)

IMPORTANTE!!
Operador . (punto)
Adapta un operador aritmético
a un operador element-wise.

Utilidad: evitar uso indebido de bucles for.

Funciones Gráficas 2D

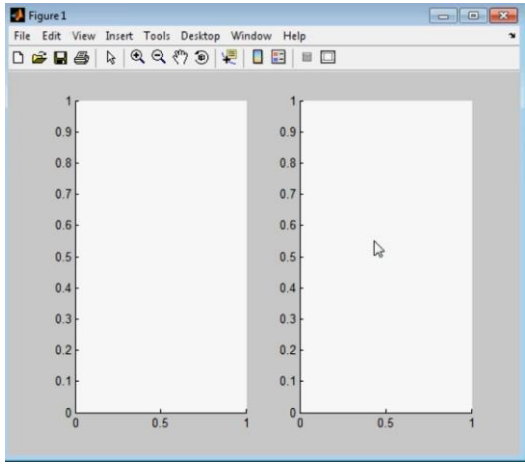
- figure(n): Genera una nueva figura (con índice n). Si la figura ya existía, esta notación nos sirve para referirnos a esta.

- subplot(f,c,n): Permite representar varias gráficas en una misma figura. Genera dentro de la figura activa una matriz de "lienzos" con **f**=filas y **c**=columnas, activando la número **n**.

- hold on: Permite pintar varios gráficos en la misma figura/lienzo.
- hold off: Desactiva el "hold". El nuevo gráfico machacará lo anterior.

- grid: Activa una cuadrícula en el dibujo.

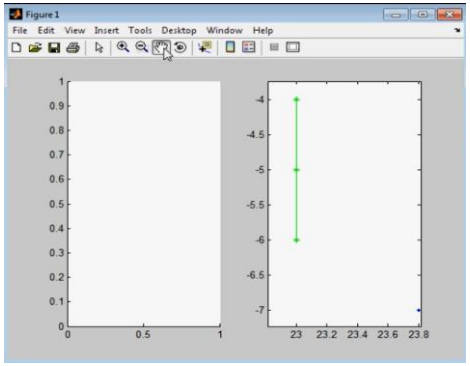
```
>> figure();
>> subplot(1,2,1);
```



Funciones Gráficas 2D

- **plot()** crea un gráfico 2D a partir de vectores con escalas lineales sobre ambos ejes.
- plot(X, Y, 'opción') La cadena de texto 'opción' permite elegir color y trazo de la curva (see help plot)

>>figure();	Genera una nueva figura
>>plot(2.3,5,'b.')	Dibuja un punto en la posición (2.3, 5) como un punto azul
>>plot(5,-2,'r^')	Dibuja un punto en la posición (5, -2) como un triangulo rojo
>>plot([2 2 2], [-6 -5 -4],'g-*')	Dibuja una linea vertical de color verde continua y marcando los puntos como asteriscos.



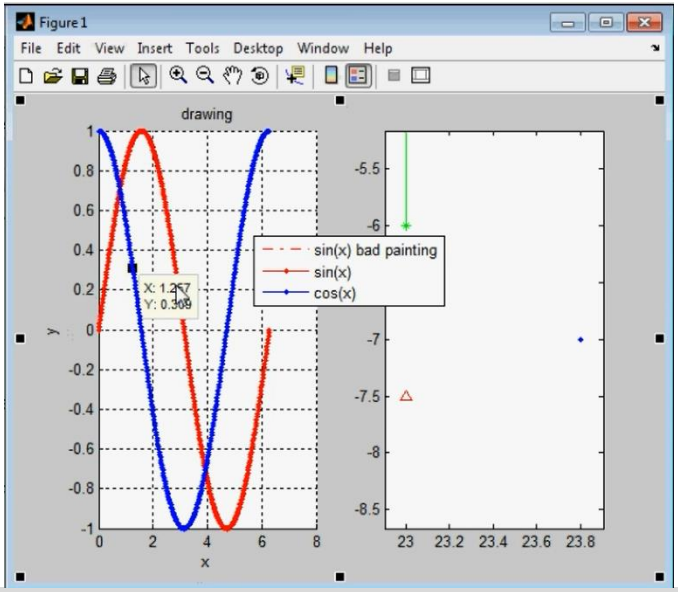
- **Otras escalas:** loglog() escala logarítmica en ambos ejes, semilogx(): escala lineal en el eje de ordenadas y logarítmica en el eje de abscisas, semilogy(): escala lineal en abscisas y logarítmica en ordenadas.

Funciones Gráficas 2D

- title('título') añade un título al dibujo.

- xlabel(‘texto’) añade una etiqueta al eje de abscisas. Con **xlabel off** desaparece.
- ylabel(‘texto’) añade una etiqueta al eje de ordenadas.
-
- legend() Genera una leyenda, es decir, asigna un nombre a cáda gráfica ploteada.
  >>legend(‘sin(x)’, ‘cos(x)’); →Asigna un texto por orden de plot.

- text(x,y,'texto') introduce 'texto' en el lugar especificado por las coordenadas **x, y**



Funciones Gráficas 2D

El comando **plot(x,y)** genera una gráfica de líneas que conecta los puntos representados por los vectores. Podemos seleccionar otros tipos de línea y/o de puntos. La siguiente tabla muestra algunas opciones:

Tipo de línea	Indicador	Tipo de punto	Indicador
continua	-	punto	.
guiones	--	más	+
punteada	:	estrella	*
guiones-puntos	-.	círculo	o
		marca	x

Resolución de Sistemas de Ecuaciones Lineales

Sistemas del tipo A*x = B

- Solución NO eficiente pero intuitiva.

  x = inv(A)*B

- Solución eficiente (recomendada)

  x = A\B

  - Si A es una matriz cuadrada de n-por-n y B es una matriz de n filas, entonces x = A\B es una solución a la ecuación A*x = B, si existe.
  - Si A es una matriz rectangular de m-por-n con m ~= n, y B es una matriz con m filas (sistema subdeterminado), entonces, A\B devuelve una solución de mínimos cuadrados al sistema de ecuaciones A*x=B.

Symbolic ToolBox

Permite hacer cálculos simbólicos y definir expresiones en Matlab.

- syms var: Util para definir variables simbólicas.

```
>>syms x                % declara x como var simbólica
>>syms x y z
```

- pretty(s): representa de forma "bonita" la expresión simbólica s

```
>> s = sym(x*(x-1)^2+x^3);
>>pretty(s)
```

- simplify(s), expand(s), factor(s), collect(s): métodos de simplificación de expresiones.
- ezplot(exp,rango): permite plotear una expression simbólica pasandole como argumento en rango de valores de sus variables simbólicas.

```
>>ezplot(s,[-5,10])
```

Programación en Matlab

Ficheros de Matlab

Escribir código en el Command Window no es muy eficiente. Sólo lo usaremos para pequeñas cosas y/o como calculadora. Para programar códigos más elaborados **Matlab nos ofrece**: scripts, live-scripts y funciones.

- **Script/live-script:** fichero de texto plano con extensión (.m / .mlx)
  - Se construyen mediante una secuencia de comandos.
  - Permiten orientación a objetos.
  - Pueden contener funciones locales (desde v2016)

- **función:** fichero que permite definir rutinas globales.
  - Comienzan siempre con una cabecera específica:

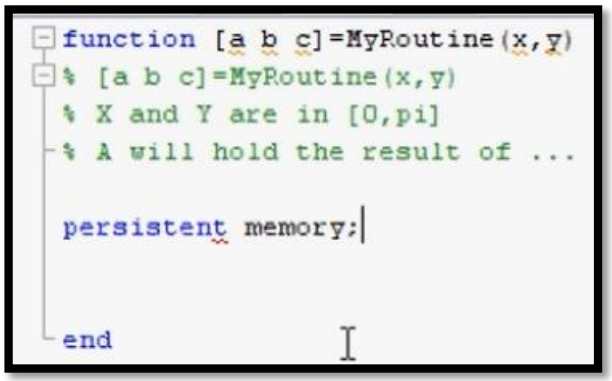
```
function arg_salida = funcion_nombre(arg_entrada, parametros)
...
end
```

  - El nombre del archivo debe coincidir con el de la función.
  - Son llamados por los ficheros de programa y/o el promt.

Programación en Matlab - Tips

- Comentarios:
  - Empiezan con el carácter **%**
  - Solo permiten 1 línea
  - ES RECOMENDABLE SU USO
- Funciones:
  - Es recomendable que las primeras líneas de una función sean comentarios explicando el sentido de los parámetros y variables, así como la funcionalidad.
  - Estos comentarios se mostrarán cuando se ejecute:

    >> help MyRoutine

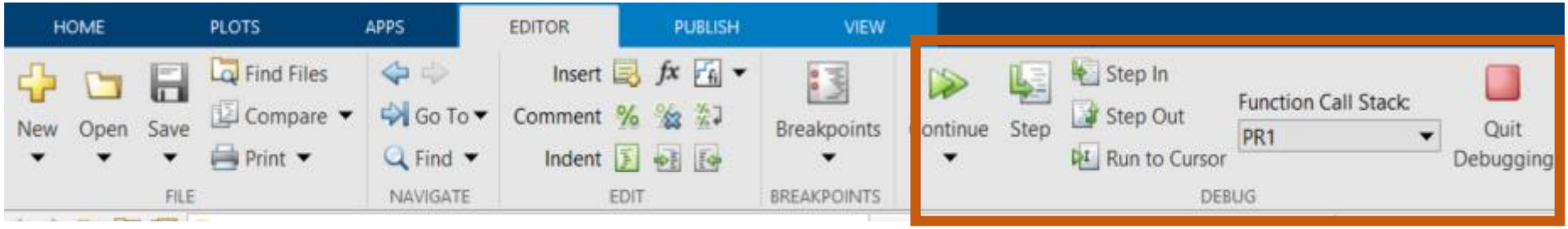


- Variables persistentes/static
  - Variables que mantienen su valor entre sucesivas llamadas a una función.
  - Es de los pocos casos en los que es necesario Declarar.

  >> persistent var_name;

Programación en Matlab - Depuración

Permite depuración paso a paso de un script. Útil para encontrar bugs y entender el funcionamiento del script.



Set/Clear breakingpoint: Coloca o borra un punto de ruptura (también con el ratón).
Clear all breakingpoints:: Borra todos los puntos de ruptura.
Step: Avanza un paso en el programa (siguiente línea de código).
Step in: Avanza un paso en el programa y si es una función, entra en dicha función.
Continue: Continua ejecutando hasta el siguiente punto de ruptura.
Quit debugging: Termina la ejecución en modo depuración.

Programación en Matlab – Bucles

Sintaxis del lenguaje de MATLAB

`if, elseif, else`	Ejecutar instrucciones si la condición es verdadera
`switch, case, otherwise`	Ejecutar uno de varios grupos de instrucciones
`for`	`for` loop para repetir el número especificado de veces
`while`	Bucle `while` para repetir cuando la condición es verdadera
`try, catch`	Execute statements and catch resulting errors
`break`	Terminar la ejecución de bucle for o while
`return`	Return control to invoking script or function
`continue`	Pass control to next iteration of `for` or `while` loop
`pause`	Stop MATLAB execution temporarily
`parfor`	Parallel for loop
`end`	Terminate block of code or indicate last array index

Programación en Matlab – Bucles

if, elseif, else

Ejecutar instrucciones si la condición es verdadera

Sintaxis

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

Operaciones lógicas:

>, <, >=,<=,== (igual)
| (or), &(and)
~ (no), ~= (no igual)

```
if r == c
    A(r,c) = 2;
elseif abs(r-c) == 1
    A(r,c) = -1;
else
    A(r,c) = 0;
end
```

```
x = 10;
minVal = 2;
maxVal = 6;

if (x >= minVal) && (x <= maxVal)
    disp('Value within specified range.')
elseif (x > maxVal)
    disp('Value exceeds maximum value.')
else
    disp('Value is below minimum value.')
end
```

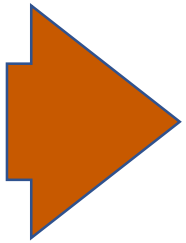
Programación en Matlab – Bucles

switch, case, otherwise

Ejecutar uno de varios grupos de instrucciones

Sintaxis

```
switch switch_expression
   case case_expression
      statements
   case case_expression
      statements
    ...
   otherwise
      statements
end
```



```
n = input('Enter a number: ');

switch n
    case -1
        disp('negative one')
    case 0
        disp('zero')
    case 1
        disp('positive one')
    otherwise
        disp('other value')
end
```

Importante: Cuando una expresión de caso es true, MATLAB ejecuta las instrucciones correspondientes a ese bloque **y sale del bloque switch.**

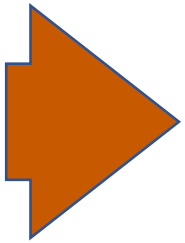
Programación en Matlab – Bucles

for

`for` loop para repetir el número especificado de veces

Sintaxis

```
for index = values
   statements
end
```



```
s = 10;
H = zeros(s);

for c = 1:s
    for r = 1:s
        H(r,c) = 1/(r+c-1);
    end
end
```

```
for v = 1.0:-0.2:0.0
   disp(v)
end
```

```
for v = [1 5 8 17]
   disp(v)
end
```

Para salir del bucle mediante programación, utilice una instrucción break. Para omitir el resto de instrucciones del bucle y comenzar la siguiente iteración, utilice continue.

Programación en Matlab – Bucles

while

Bucle `while` para repetir cuando la condición es verdadera

Sintaxis

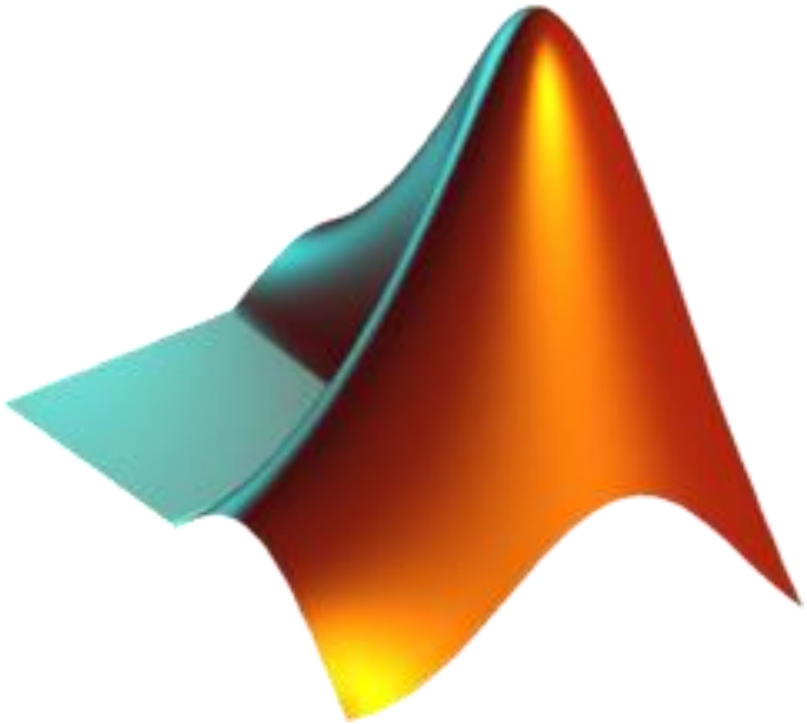
```
while expression
    statements
end
```

```
n = 10;
f = n;
while n > 1
    n = n-1;
    f = f*n;
end
disp(['n! = ' num2str(f)])
```

Cuenta el número de líneas de código del archivo magic.m. Omitiendo las líneas en blanco y los comentarios.

```
fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line) || strncmp(line,'%',1) || ~ischar(line)
        continue
    end
    count = count + 1;
end
count
```

MATLAB y Ec. Diferenciales



Ec. Diferenciales en Matlab

[t, y] = ode45(odefun, tspan, y0)

- **odefun**: Sistema de ecuaciones diferenciales a resolver, especificadas como identificador de función que define las funciones que se desea integrar. Debe ser del tipo y'=f(t,y) "solo de primer orden".
- **tspan**: [t0 tf] intervalo de tiempos para la solución, aunque también se puede pasar un vector de tiempos tipo t = 0:0.1:10;
- **y0** = condiciones iniciales de la ecuación diferencial.

Esta instrucción integra el sistema de ecuaciones diferenciales $y'=f(t,y)$ de t0 a tf con condiciones iniciales y0.

Ayuda de Matlab: ode45 es un versátil solver de ODE y es el primer solver que debería probar para solucionar la mayoría de problemas. Sin embargo, si el problema es rígido o requiere mucha precisión, existen otros solvers de ODE que podrían ser más adecuados para resolverlo. Para obtener más información, consulte Elegir un solver de ODE.

Ec. Diferenciales en Matlab (Ejemplo)

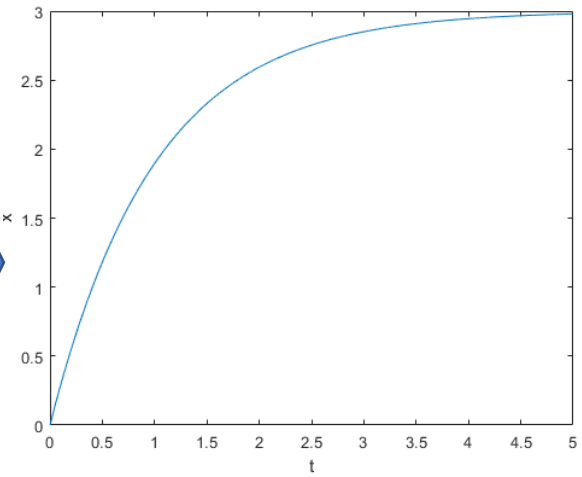
Ejemplo: Resolver y visualizar el valor de "x" durante 5 segundos.

$$\frac{dx}{dt} = 3e^{-t}$$

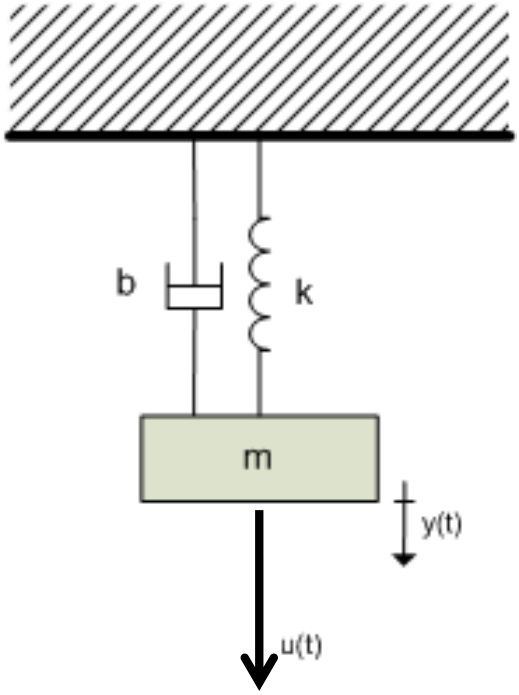
$$x(0) = 0$$

```
% SOLVE  dx/dt = -3 exp(-t).
% initial conditions: x(0) = 0

t = 0:0.001:5;    % vector de tiempos
x0 = 0;           % condicion inicial

% resolvemos la Ec. Diff
[t,x] = ode45(@(t,x) odefcn(t,x), t, x0);

% mostramos el resultado gráficamente
figure();
plot(t,x);
xlabel('t'); ylabel('x');
```

```
function dxdt = odefcn(t,x)
    dxdt = 3*exp(-t);
end
```



Ec. Diferenciales en Matlab



$$\frac{d^2x}{dt} = \frac{1}{m}\left[u(t) - b \cdot \frac{dx}{dt} - k \cdot x(t)\right]$$

Simular el comportamiento del sistema de masas traslacional de la figura, visualizando el desplazamiento de la masa "m" a lo largo del tiempo.

- Datos:
  - m = 3 Kg
  - b = 5 N·s/m,
  - k = 2 N/m,
  - f = 5 N

-¿Cuál es el desplazamiento final en el que se queda estabilizado el sistema?

Ec. Diferenciales en Matlab

$$\frac{d^2x}{dt} = \frac{1}{m}\left[u(t) - b \cdot \frac{dx}{dt} - k \cdot x(t)\right]$$

```
% Parámetros del sistema
M = 3;      % Masa de la primera masa (kg)
f = 5;      % Fuerza externa (N)
B = 5;      % Cte. amortiguacion (N·s/m)
K = 2;      % Cte. elastica (N/m)

% Condiciones iniciales
t_span = [0, 20];   % Intervalo de tiempo (s)
init_cond = [0; 0]; % Posición y velocidad de la masa para t=0
```

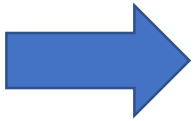
```
% Solver Ec. Diff
[t, y] = ode45(@(t, y) odefcn(t,y,M,f,B,K), t_span, init_cond);
```

Ec. Diferenciales en Matlab

@odefcn debe ser del tipo y'=f(t,y) "solo de primer orden"

$$\frac{d^2x}{dt} = \frac{1}{m}\left[u(t) - b \cdot \frac{dx}{dt} - k \cdot x(t)\right]$$

Ecuación de segundo orden



$$\frac{d\,x}{dt} = y$$

$$\frac{d\,y}{dt} = \frac{1}{m}\left[u(t) - b \cdot y - k \cdot x(t)\right]$$

Sistema de 2 ecuaciones de primer orden

Ec. Diferenciales en Matlab

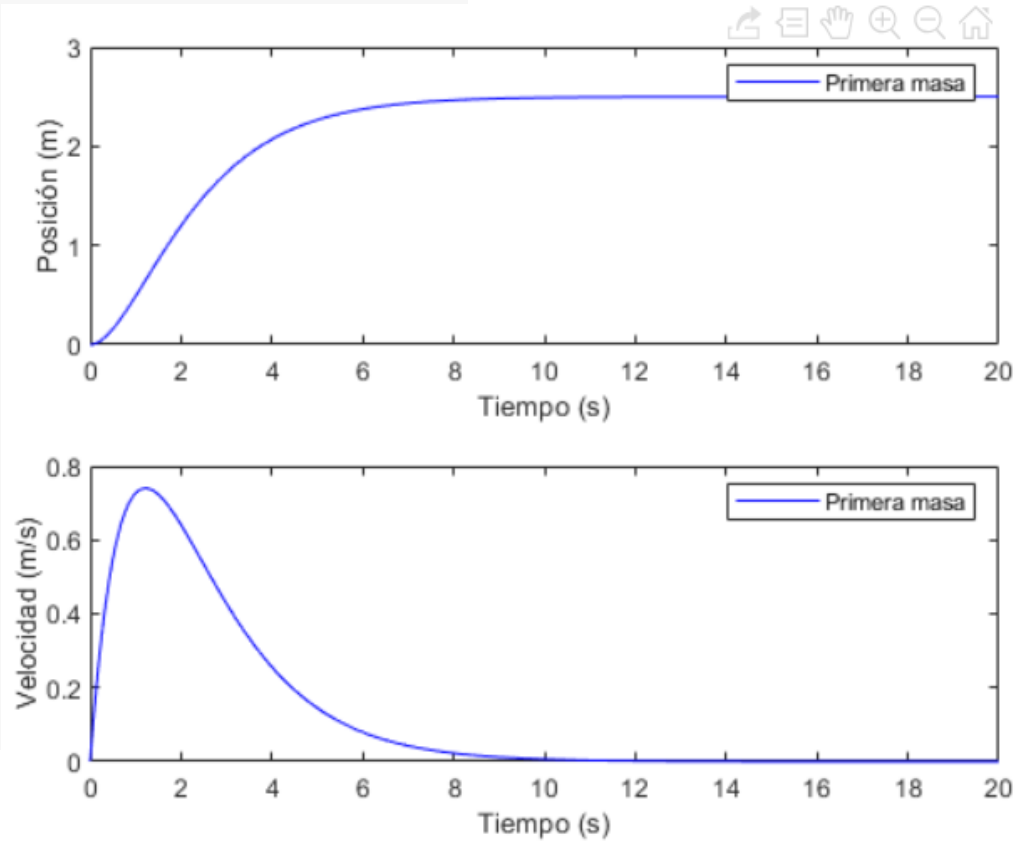
$$\frac{dx}{dt} = y$$

$$\frac{dy}{dt} = \frac{1}{m}\left[u(t) - b \cdot y - k \cdot x(t)\right]$$

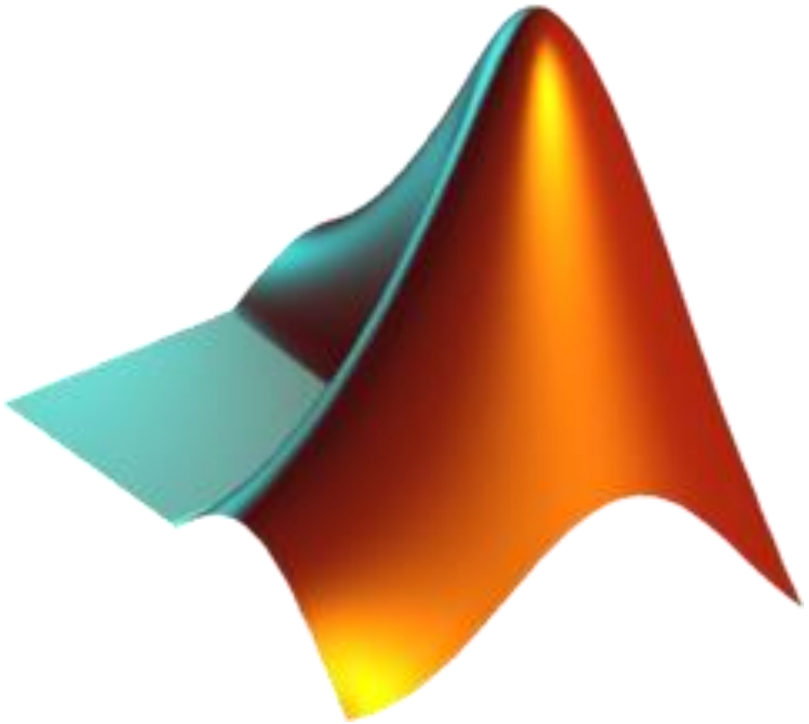
```
function dydt = odefcn(t,y,M,f,B,K)
    % Ecuación diferencial del sistema
    % ode45 solo resuleve ec. diff primer orden
    % por ello transformamos nuestra ediff de 2º orden en
    % dos ediff de primer orden (parecido a un cambio de variable)
    % y(1)=x (posición), y(2)=dx/dt (velocidad)

    % dydt(1) = dx/dt = y(2)
    % dydt(2) = f/M - B/M*y(2) - K/M*y(1)
    dydt = [y(2); f/M - B/M*y(2) - K/M*y(1)];
end
```

Ec. Diferenciales en Matlab

```
% Solver Ec. Diff
[t, y] = ode45(@(t, y) odefcn(t,y,M,f,B,K), t_span, init_cond);

% Extracción de resultados
x_pos = y(:, 1);
x_vel = y(:, 2);

% Visualización de resultados
figure;
grid on;
subplot(2, 1, 1);
plot(t, x_pos, 'b-');
xlabel('Tiempo (s)');
ylabel('Posición (m)');
legend('Primera masa');
subplot(2, 1, 2);
plot(t, x_vel, 'b-');
xlabel('Tiempo (s)');
ylabel('Velocidad (m/s)');
legend('Primera masa');
```

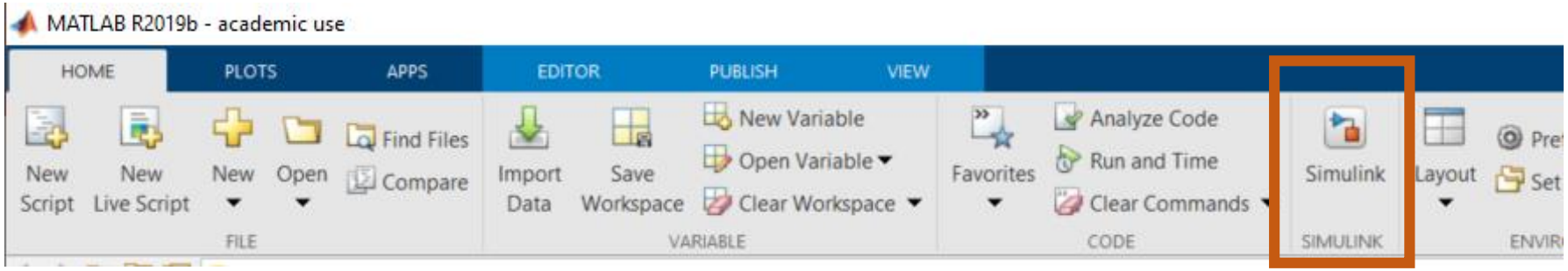


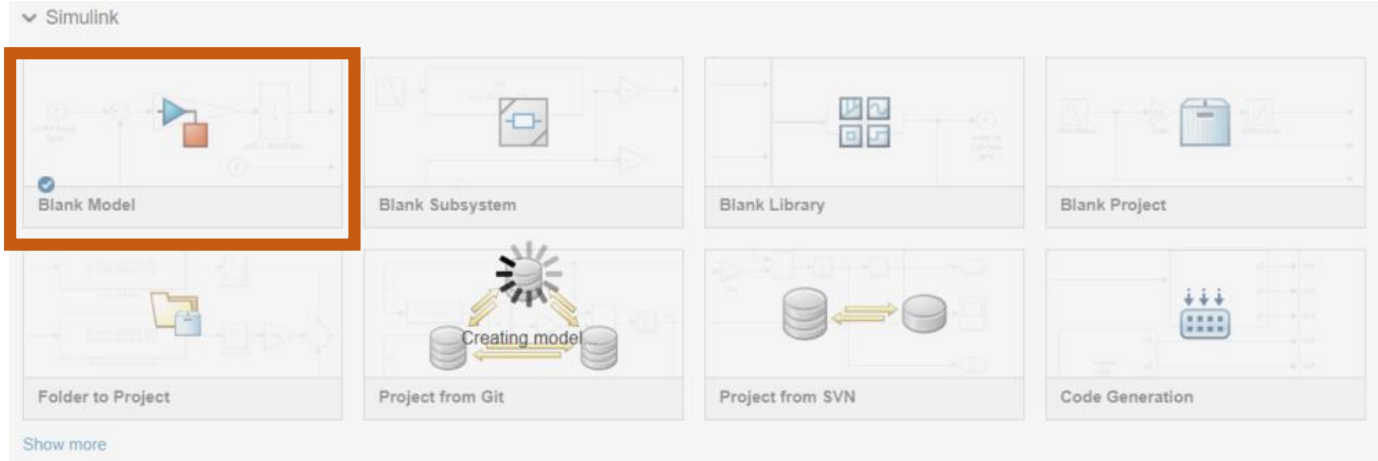
Introducción a SIMULINK



SIMULINK

- Es un toolbox de Matlab.
- Ofrece programación gráfica y simulación de sistemas.

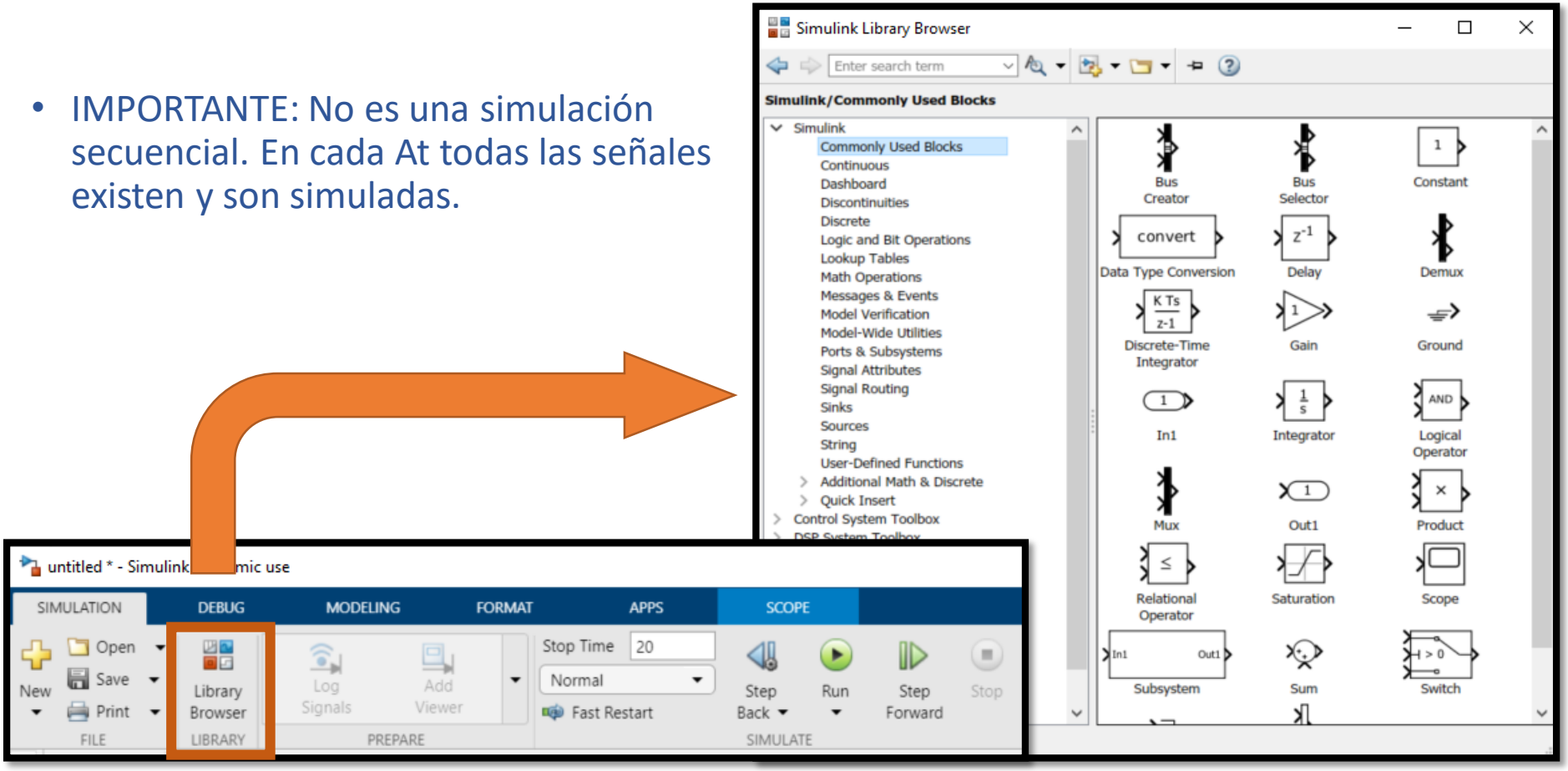




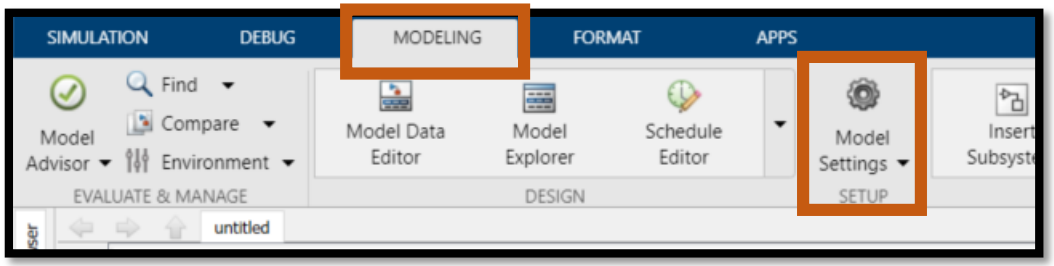
SIMULINK

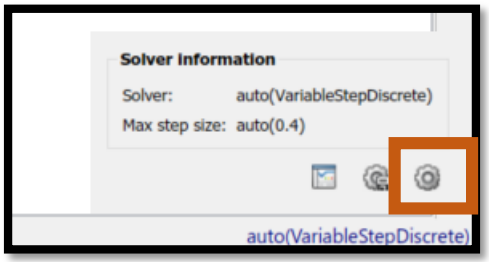
- Simulink es un entorno gráfico basado en diagramas de bloques para simulación multidominio y diseño basado en modelos.

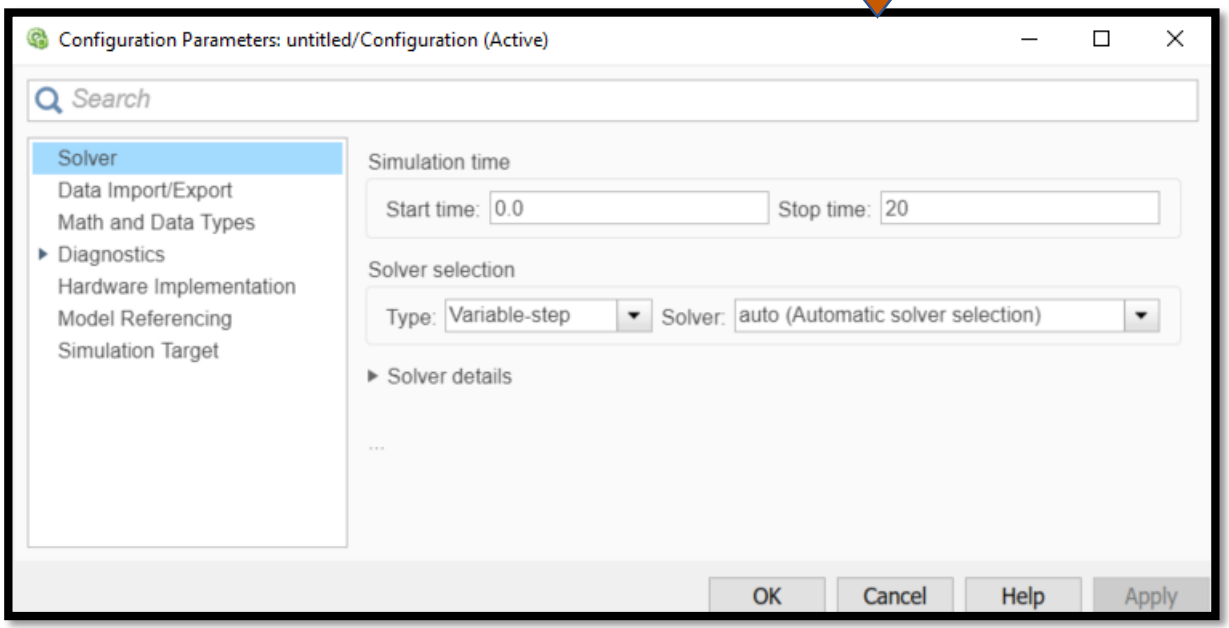
- IMPORTANTE: No es una simulación secuencial. En cada At todas las señales existen y son simuladas.



SIMULINK – Configuración de Simulación

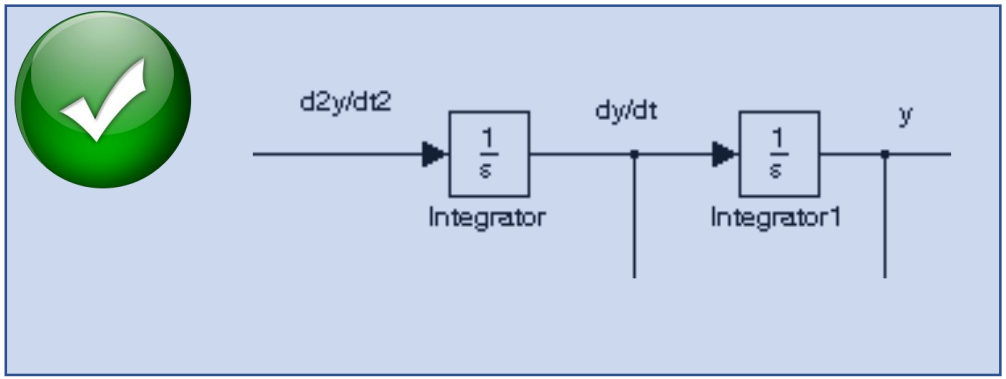


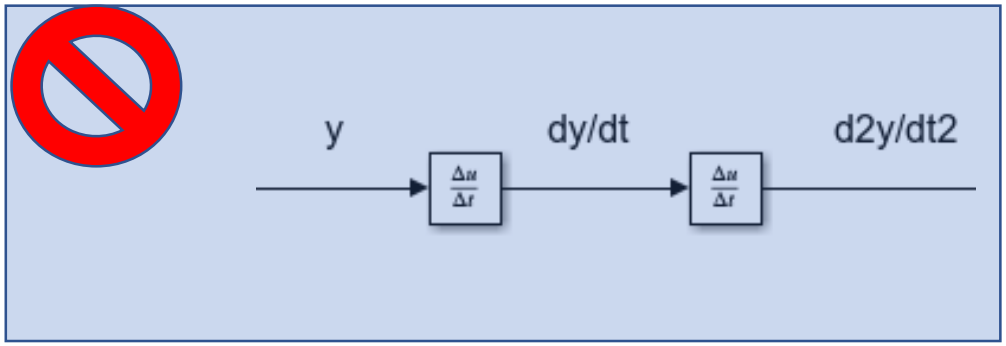




SIMULINK – Ecuaciones Diferenciales

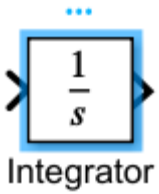
- Para modelar Ec. diferenciales usar integradores → No usar derivadores pues estos amplifican el ruido.





SIMULINK – Elementos Básicos

Ganancia (puede ser una variable)



Integrador (puede definir su condición inicial)

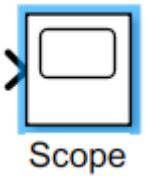


Gráfico 2D (eje X es el tiempo)

Suma de señales (se puede aumentar el número de señales, y hacer restas)

Genera un dato Cte.

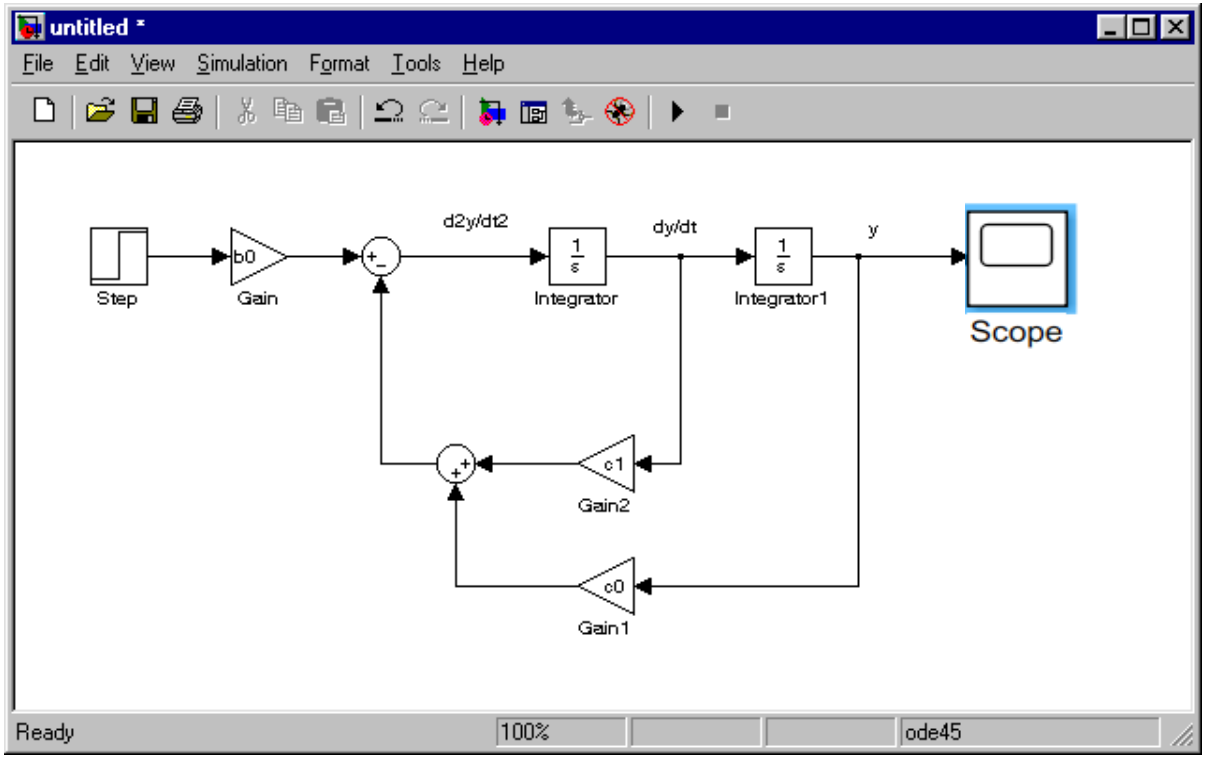
Genera un señal “escalón”

SIMULINK – Ejemplo 1

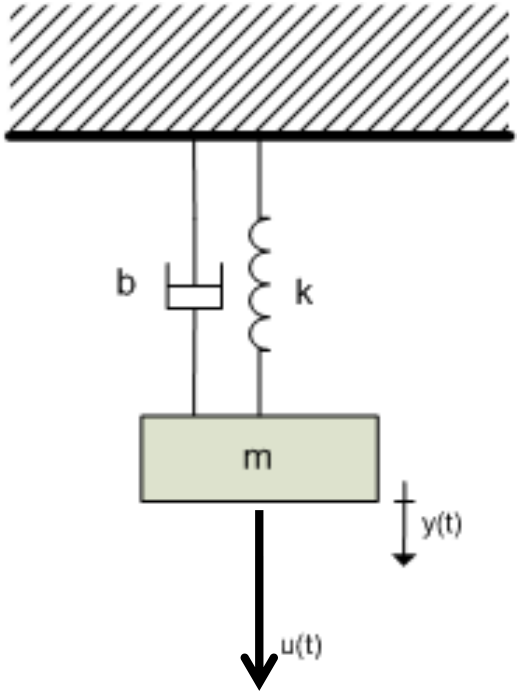
Simular el comportamiento de un sistema caracterizado por la siguiente ecuación diferencial ante la respuesta al escalón unidad.

$$\frac{d^2y}{dt^2} + c_1 \frac{dy}{dt} + c_0 y = b_0 f(t)$$

$$\frac{d^2y}{dt^2} = b_0 f(t) - c_1 \frac{dy}{dt} - c_0 y$$



SIMULINK – Ejemplo 2



$$m \cdot \frac{d^2y}{dt} + b \cdot \frac{dy}{dt} + k \cdot y(t) = u(t)$$

$$\frac{d^2y}{dt} = \frac{1}{m}\left[u(t) - b \cdot \frac{dy}{dt} - k \cdot y(t)\right]$$

Simular el comportamiento del sistema de masas traslacional de la figura, visualizando el desplazamiento de la masa "m" a lo largo del tiempo.

- Datos:
  - m = 10 Kg
  - b = 1.8 Ns/m,
  - k = 23 N/m,
  - f = 72 N

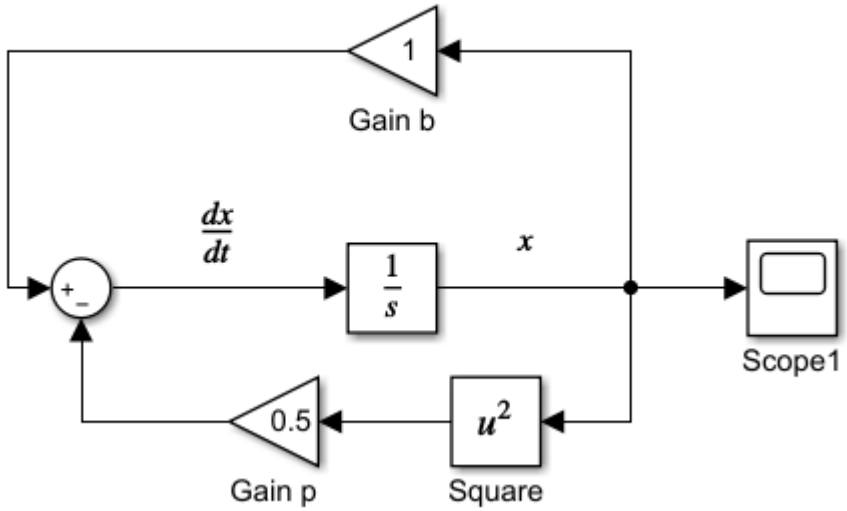
-¿Cuál es el desplazamiento final en el que se queda estabilizado el sistema?

SIMULINK – Ejemplo 3

Simular el comportamiento del sistema descrito por la siguiente ecuación diferencial:

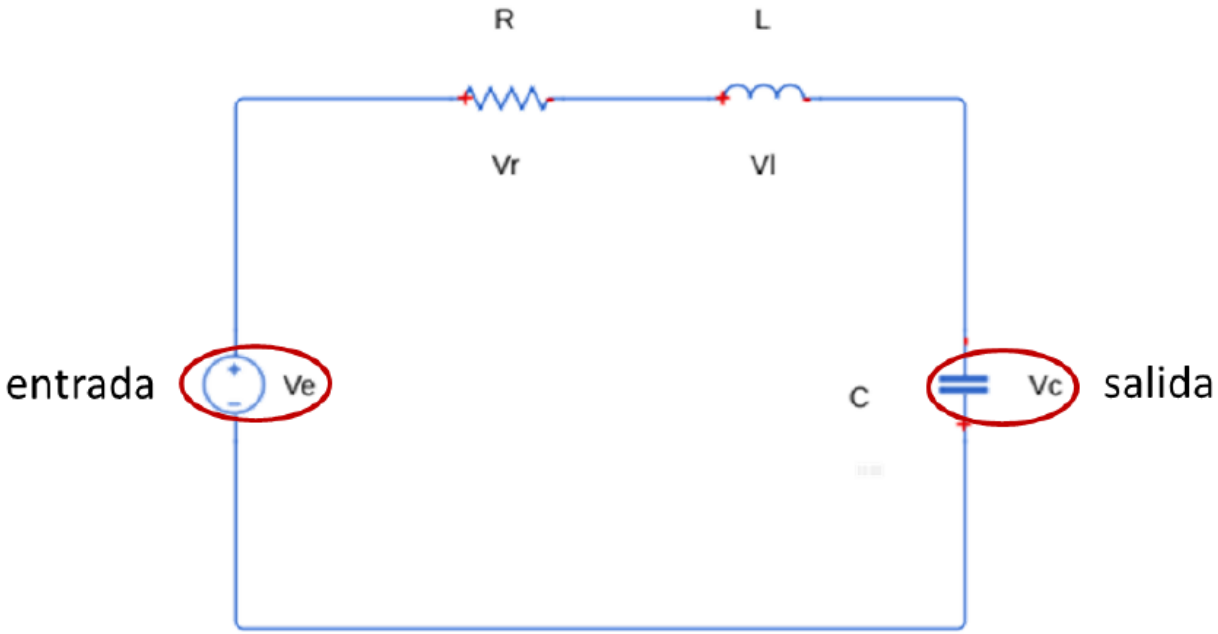
$$\frac{d\,x}{dt} = bx\ - px^2$$

- **Datos**:
  - b = 1
  - p = 0.5
  - x(0) = 100

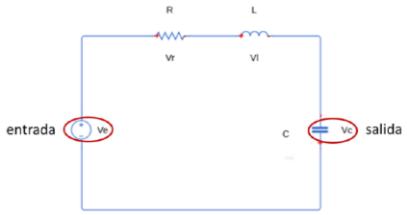


SIMULINK – Ejemplo Propuesto

P1. Para el circuito RLC de la figura, implementa su modelo Simulink y simúlalo para una entrada de $12\,V, R = 1, L = 3, C = 2$.

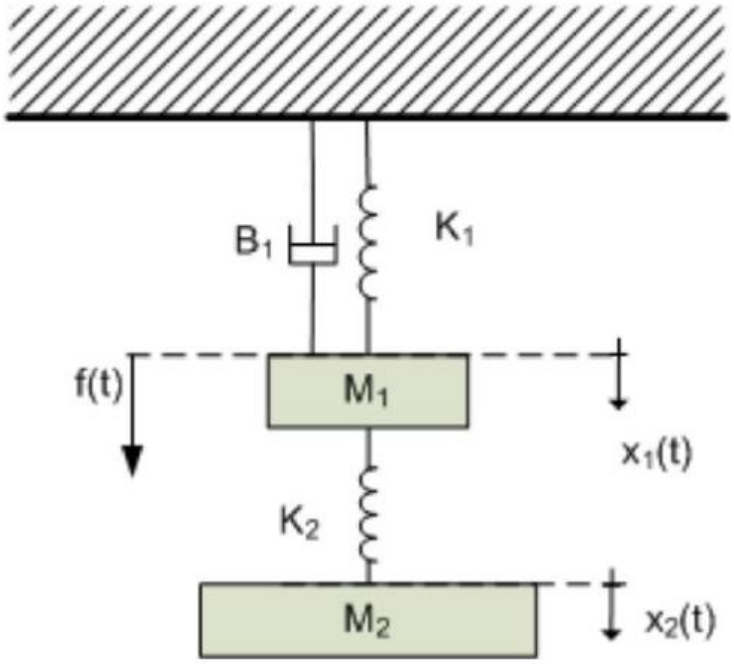


SIMULINK – Ejemplo Propuesto

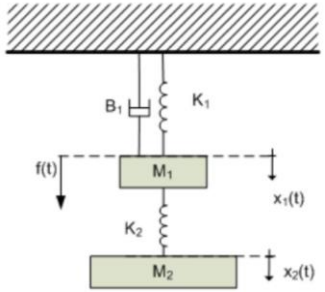


SIMULINK – Ejemplo Propuesto

P2. Para el sistema de masas de la figura, obtén las ecuaciones diferenciales y modélalas en SIMULINK. Para valores de $B_1 = 5, K_1 = 2, K_2 = 3, M_1 = 2, M_2 = 0.1$ simula y muestra las posiciones x_1 y x_2 para una entrada escalón de 5 N.

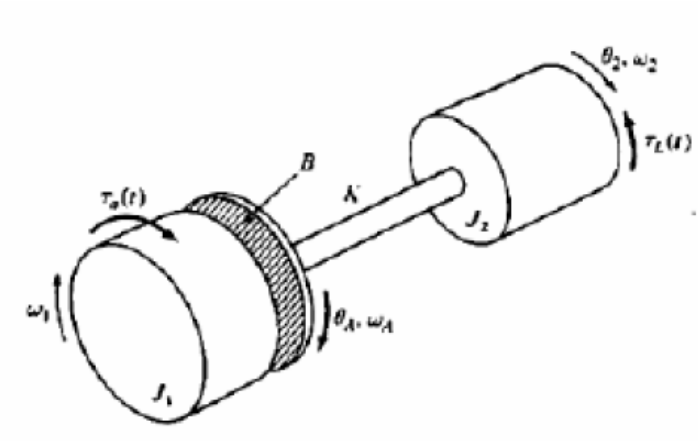


SIMULINK – Ejemplo Propuesto



SIMULINK – Ejemplo Propuesto

P3. El sistema rotacional de la figura se compone de dos masas interconectadas por un eje que se opone tanto a sus posiciones relativas (muelle rotacional), como a sus velocidades relativas (amortiguador rotacional). Además la inercia J_1 puede ser girada, es decir, recibir una entrada de torsión. Modelar y simular el sistema ante señales de entrada a la inercia J_1 de $u = sen(t)$, mostrando la velocidad y la posición angular de ambas inercias. Considera $J_1 = 1, J_2 = 0.5, K = 3, B = 2$.



SIMULINK – Ejemplo Propuesto