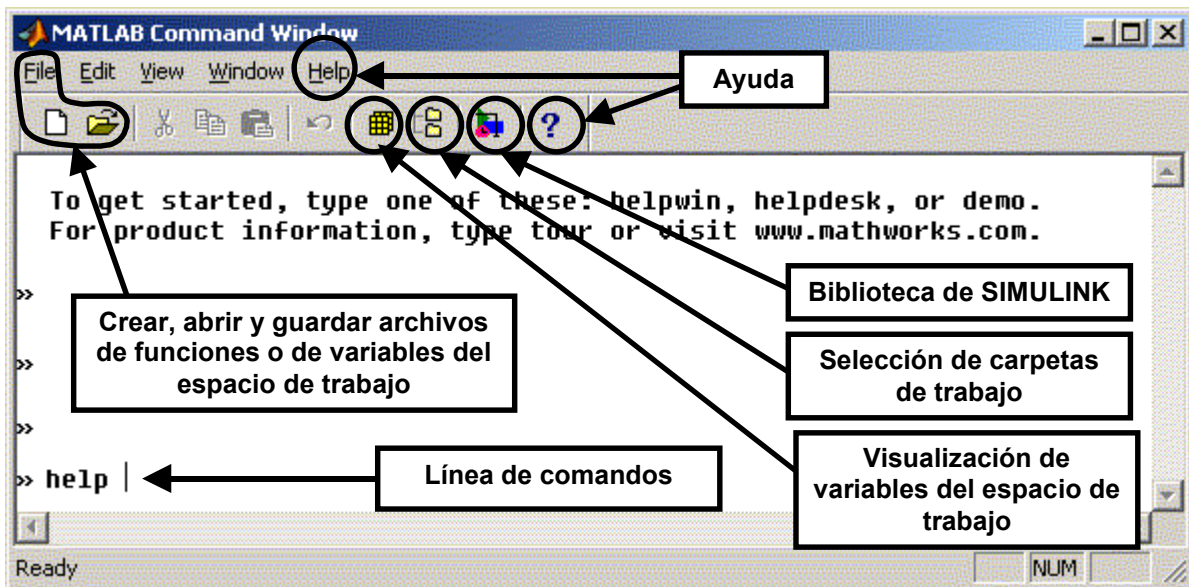


Simulación de sistemas de control continuos con MATLAB y SIMULINK





Comandos básicos de MATLAB para las prácticas de Regulación

1. Comandos básicos. Variables, vectores y matrices.

Matlab es un programa de cálculo matemático muy flexible y potente, con posibilidades gráficas para la presentación de los datos, por lo que se utiliza en muchos campos de la ciencia y la investigación como herramienta de cálculo matemático. Los comandos y funciones que se describen a continuación funcionan generalmente en todas las versiones de MATLAB, aunque algunos sólo lo hacen en las versiones más modernas y otros han quedado obsoletos. En este último caso las versiones más modernas disponen de otros comandos o funciones que los sustituyen. Algunos de ellos están disponibles bajo botones o menús en las nuevas versiones.

NOTA: En este documento se recogen unos pocos ejemplos de entre las inmensas posibilidades que ofrece Matlab. Se deben consultar las ayudas de Matlab u otros documentos si se desea un conocimiento más amplio. Se deben utilizar y modificar los ejemplos aquí incluidos para adquirir las destrezas básicas que permitirán utilizar el programa como herramienta para la asignatura. Téngase en cuenta que algunos ejemplos están encadenados, es decir, es necesario ejecutar determinados comandos anteriores para que los siguientes funcionen correctamente.

Realizar operaciones algebraicas es muy sencillo, o simplemente:

```
>> a = 2
>> b = 3
>> suma = a + b
suma =
    5
```

```
>> 2 + 3
ans =
    5
```

La variable “ans” contiene el resultado de la última operación realizada y puede consultarse en cualquier momento así como el resto de las variables que se vayan creando:

```
>> suma
suma =
    5
```

```
>> ans
ans =
    5
```

Si se quiere ver el nombre de todas las variables que se están utilizando se pueden usar los comandos **who** y **whos**. Con **save** y **load** se pueden guardar las variables que se están usando y recuperarlas posteriormente. El comando **help** proporciona ayuda y combinado con otro comando, ayuda sobre ese comando en concreto (p.ej. **help who**). Para abandonar el programa se pueden usar los comandos **quit** o **exit**. (En versiones bajo Windows se dispone de los correspondientes menús y botones para estas funciones)

Matlab maneja matrices, y como casos particulares de las mismas, vectores fila y vectores columna:

```
>> matriz = [1 2 3
4 5 6
7 8 9]
>> matriz = [1,2,3;4,5,6;7,8,9]
```

```
>> vectorfila = [1 2 3]
>> vectorfila = [1,2,3]
>> vectorcolumna = [1,2,3]' (' transposición)
>> vectorcolumna = [1;2;3]
```

2. Valores complejos.

```
>> c = 2 + 3*j
>> modulo = abs(c)
```

-> se utilizan “j” o “i” como valores imaginarios
-> se obtiene el módulo

>> **argumento = angle(c)** -> el valor del argumento esta en radianes
>> **grados = argumento * 180 / pi** -> pero se convierte fácilmente a grados

NOTA: obviamente "i", "j", "ans", "pi", "help", "sin", etc. son nombres que ya están definidos para variables, constantes, comandos o funciones de Matlab y no se deben usar para nombrar nuevas funciones o variables del usuario. ¡Matlab distingue entre mayúsculas y minúsculas!

3. Polinomios.

Los polinomios se introducen en forma de vectores fila que contienen los coeficientes del polinomio.

>> **p = [1 -6 0 -27]** -> $s^3 - 6s^2 - 27$
>> **raices = roots(p)** -> da las raíces del polinomio en un vector columna
>> **polinomio = poly(raices)** -> se vuelve a obtener el polinomio original

Se pueden multiplicar y dividir polinomios: $c(s) = a(s) * b(s)$; $c(s)/a(s) = q(s) + r(s)/a(s)$

>> **a = [1 2 3]** -> $s^2 + 2s + 3$
>> **b = [4 5 6]** -> $4s^2 + 5s + 6$
>> **c = conv(a,b)**
c =
4 13 28 27 18 -> producto de los dos polinomios = $4s^4 + 13s^3 + 28s^2 + 27s + 18$
>> **[q,r] = deconv(c,a)** -> división de $c(s)$ por $a(s)$
q =
4 5 6 -> $q(s)$ es el cociente de la división
r =
0 0 0 0 0 -> $r(s)$ es el resto de la división

Como se puede observar, a veces el resultado de una función, pueden ser varias variables y sus argumentos también pueden ser varios y de diferentes tipos. Se debe consultar mediante el comando **help** (p.ej. **help deconv**) cuales son los argumentos que admite cada función y que variables va a devolver como resultado.

4. Representaciones gráficas 2D y 3D.

Representación de la función seno:

>> **t = [0:1:100]** -> vector tiempo
>> **x = t** -> eje x
>> **y = sin(0.1*t)** -> eje y
>> **plot(x,y)**

Representación en 3D:

>> **t = [0:1:100]** -> vector tiempo
>> **x = t** -> eje x
>> **y = sin(0.1*t)** -> eje y
>> **z = cos(0.2*t)** -> eje z
>> **plot3(x,y,z)**

Representación de dos funciones:

>> **t = [0:1:100]**
>> **x = t**
>> **y = [sin(0.1*t) ; cos(0.1*t)]**
>> **plot(x,y)**

5. Funciones de Transferencia.

Para trabajar con funciones de transferencia, se introducen por separado dos vectores, uno con los coeficientes del polinomio del numerador y otro con los del denominador:

```
>> num = [1 2]           -> s+2
>> den = [1 3 5]        -> s2+3s+5
>> printsys(num,den)    -> muestra la función de transferencia
    (s+2)/(s2+3s+5)
>> pzmap(num,den)       -> muestra el mapa de ceros y polos del sistema gráficamente
>> [p,z] = pzmap(num,den) -> en los vectores columna "p" y "z" se tendrán los polos y
                                ceros respectivamente del sistema
>> pzmap(p,z)           -> muestra el mapa de ceros y polos del sistema a partir de estos
```

Para un sistema de segundo orden:

```
>> wn = 30               -> valor para  $\omega_n$ 
>> xi = 0.3              -> valor para  $\xi$ 
>> [num2,den2] = ord2(wn,xi) -> función de transferencia de segundo orden con  $\omega_n$  y  $\xi$ 
>> printsys(num2,den2)   -> la función de transferencia de segundo orden
    1/(s2+18s+900)
```

En versiones de MATLAB superiores a la 5.0 se puede trabajar con el objeto especial Función de Transferencia, que permite operar cómodamente con ellas ...

```
>> g = tf([1 2], [1 3 5]) -> función "tf"
    (s+2)/(s2+3s+5)
```

... o incluso representarlas y operar con ellas en forma simbólica:

```
>> s = tf('s')          -> crea el objeto "s"
>> g = (s+2)/(s2+3*s+5)
    (s+2)/(s2+3s+5)
>> g2 = g*g
```

Estos objetos Función de Transferencia se pueden sumar, multiplicar, etc. y aplicarles las funciones que aquí se describen, válidas también para un par de polinomios numerador/denominador:

```
>> num = [1 2]           -> s+2
>> den = [1 3 5]        -> s2+3s+5
>> pzmap(num,den)

Equivale a:
>> g = tf([1 2], [1 3 5]) -> g=(s+2)/(s2+3s+5)
>> pzmap(g)
```

Y para pasar de 'num/den' a 'g' y de 'g' a 'num/den':

```
>> g = tf(num,den)
>> [p,z] = pzmap(g)
>> num = dcgain(g*tf(poly(p),poly(z)))*poly(z)
>> den = poly(p)

O más directamente:
>> g = tf(num,den)
>> num = deal(g.num{:})
>> den = deal(g.den{:})
```

```
>> m = feedback(g,1,-1) -> proporciona la función de transferencia en bucle cerrado con
                                realimentación unitaria "1" y negativa "-1"
```

6. Respuesta en el tiempo.

```
>> impulse(g)            -> representa gráficamente la respuesta en el tiempo del sistema
                                ante una entrada impulso
>> step(g)               -> representa gráficamente la respuesta en el tiempo del sistema
                                ante una entrada escalón unitario. Un clic con el ratón sobre
                                las gráficas proporciona información sobre sus valores
>> [y,t] = step(g)       -> almacena en "y" la respuesta del sistema y en "t" el vector de
                                tiempo.
>> [maximo,indice] = max(y) -> da el valor máximo que alcanza la respuesta del sistema
>> t(indice)              -> instante en el que se produce el valor máximo
```



7. Respuesta en frecuencia.

- >> bode(g)** -> dibuja el diagrama de bode del sistema
- >> [mag,fase] = bode(g,3)** -> da los valores de magnitud (no está en decibelios) y de fase (en grados) del sistema para $\omega = 3$
- >> [mag,fase,w] = bode(g)** -> devuelve los valores de magnitud, fase y pulsación, en tres vectores, para utilizarlos posteriormente con otras funciones como por ejemplo:
- >> maximo = max(mag)** -> valor en el pico de resonancia de la relación de amplitudes
- >> resonancia = 20*log10(maximo)** -> valor en el pico de resonancia expresado en decibelios
- >> [Gm,Pm,wg,wp] = margin(g)** -> \lll Resultados extraños ???
Indica que el margen de fase y de ganancia son infinito, y que no se puede determinar un valor de ω para ellos (NaN = Not a Number)
- Gm = ∞**
- Pm = ∞**
- wg = NaN**
- wp = NaN**
- >> g1 = 10/((s+1)*(s+2)*(s+3))** -> se crea una nueva FdT para probar la función "margin"
- >> [Gm1,Pm1,wg1,wp1] = margin(g1)**
- Gm1 =** -> margen de ganancia (no está en decibelios)
- Pm1 =** -> margen de fase en grados
- wg1 =** -> pulsación (frecuencia en rad/s) para el margen de ganancia
- wp1 =** -> pulsación (frecuencia en rad/s) para el margen de fase
- >> margin(g1)** -> representa gráficamente el margen de ganancia y de fase
- >> nichols(g)** -> diagrama magnitud-fase
- >> ngrid('new')** -> superpone el ábaco de Nichols al diagrama anterior
- >> shg** -> muestra la pantalla gráfica
- >> clf** -> limpia la pantalla gráfica (en versiones antiguas "clg")
- >> hold on** -> bloquea la pantalla gráfica para superponer un nuevo trazado
- >> hold off** -> desbloquea la pantalla gráfica para realizar nuevos trazados
- >> nyquist(g)** -> diagrama de Nyquist del sistema. El tramo con valores de $\omega > 0$ es el "Diagrama Polar" del sistema
- >> ltiview(g)** -> permite representar conjuntamente las gráficas ya descritas para varios sistemas simultáneamente

8. Lugar de las raíces.

- >> rlocus(g)** -> lugar de las raíces del sistema
- >> polos = rlocus(num,den,3)** -> polos del sistema en bucle cerrado para "k=3"
- >> hold on** -> fija la pantalla gráfica para seguir trabajando sobre ella
- >> rlocus(num,den,[3 4])** -> dibuja la posición de las raíces (polos) del sistema en bucle cerrado para los valores de "k=3" y "k=4"
- >> rltool** -> herramienta para el diseño de reguladores basado en el L.R.
- >> sisotool** -> herramienta de diseño más genérica que "rltool"

NOTA: Las unidades de los resultados (grados, radianes, decibelios, etc.) pueden variar según las versiones de Matlab.

Elementos básicos de SIMULINK

1. La biblioteca de Simulink.

Simulink proporciona un entorno gráfico al usuario que facilita enormemente el análisis, diseño y simulación de sistemas (de control, electrónicos, etc.), al incluir una serie de rutinas que resuelven los cálculos matemáticos de fondo, junto con una sencilla interfaz para su uso. Proporciona un entorno de usuario gráfico que permite dibujar los sistemas como diagramas de bloques tal y como se haría sobre un papel.

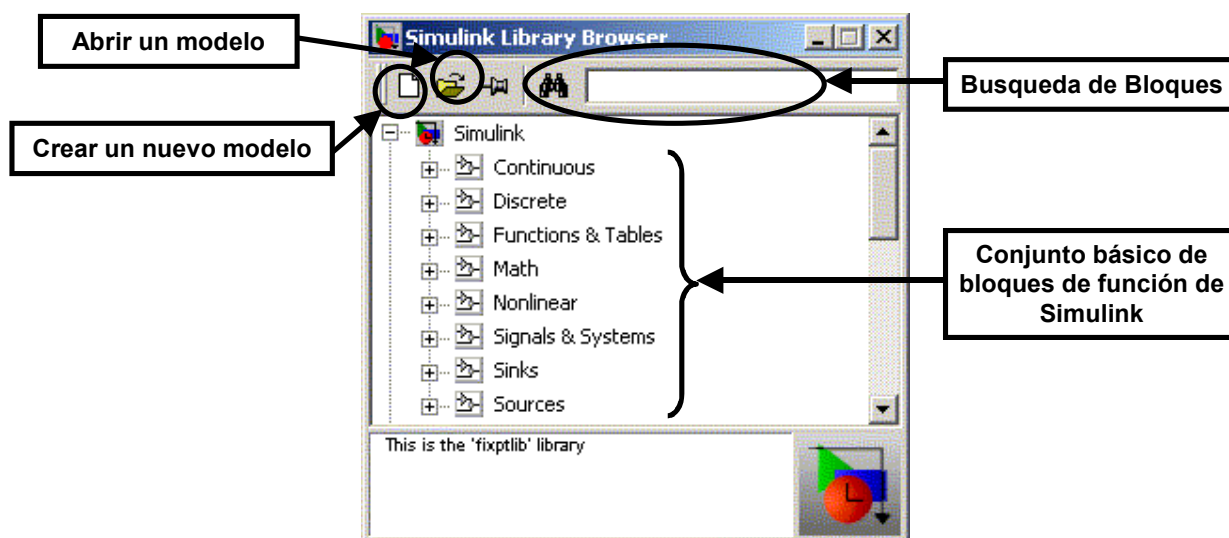
El conjunto de componentes incluidos junto al programa Simulink, incluye bibliotecas de fuentes de señal, dispositivos de presentación de datos, sistemas lineales y no lineales, conectores y funciones matemáticas. En caso de que sea necesario, se pueden crear nuevos bloques a medida por el usuario.

El programa Simulink se inicia desde el botón "Simulink Library Browser" (Biblioteca de Simulink, ver imagen de portada) de la ventana de comandos de Matlab, o desde la línea de comandos mediante la orden:

```
>> simulink
```

Una vez iniciado el programa el entorno de trabajo queda dividido en tres partes.

- La ventana de comandos de Matlab (*Matlab command window*): desde la que se puede ejecutar cualquier comando del mismo, dar valores a variables y controlar la ejecución de las simulaciones.



- La ventana de la biblioteca de Simulink (*Simulink Library Browser*): desde la que se seleccionan los componentes que se van a insertar en el sistema a simular.
- La o las ventanas de los modelos: en las que se dibujan los modelos y se realizan y controlan las simulaciones. Estas ventanas aparecen cuando se abre un modelo ya existente o se crea una ventana en blanco para dibujar un nuevo modelo. Para ello, se pueden utilizar los botones de la ventana de la librería de Simulink.

Todos los componentes básicos de Simulink, se pueden encontrar en su biblioteca de componentes. A continuación se describen los componentes básicos de la biblioteca de Simulink:

Continuous: Bloques que pueden ser representados como una función continua en el tiempo.

Derivative: La salida del bloque se corresponde con la derivada de la entrada.

Integrator: La salida del bloque se corresponde con la integral de la entrada. Los parámetros del bloque permiten controlar el valor inicial de la salida, así como la existencia de límites superiores e inferiores en la salida.

Transfer Fcn: Permite expresar una función de transferencia mediante su expresión en la variable compleja s . Sus parámetros son los polinomios del numerador y del denominador de la función, expresados como vectores fila.

Transport Delay: La salida del bloque se corresponde con la entrada al mismo retrasada una cantidad de tiempo, que se fija como parámetro en el bloque.

Zero-Pole: Función de transferencia expresada en función de la ganancia en régimen permanente, y la situación de los polos y ceros del sistema.

Math: Bloques que realizan operaciones matemáticas sobre sus entradas.

Abs: Calcula el valor absoluto de su entrada.

Gain: Aplica una ganancia constante a la entrada.

Math Function: Este bloque incluye la mayor parte de las funciones matemáticas típicas, con la excepción de las funciones trigonométricas.

Product: Calcula el producto escalar de sus entradas. Un parámetro del bloque permite regular el número de entradas del mismo.

Sign: Calcula el signo de la entrada. +1 indica positivo, -1 negativo, y 0 un valor nulo.

Sum: Calcula la suma de todas sus entradas. Un parámetro permite indicar el número de entradas, y si estas deben ser invertidas antes de la suma. Ejemplo: un valor para el parámetro "+++" indicaría que el bloque tiene 4 entradas, y que la tercera de ellas debe ser invertida antes de sumarla.

Trigonometric Function: En este bloque se incluyen todas las funciones trigonométricas típicas.

Nonlinear: Bloques no lineales.

Dead Zone: Incluye una *zona muerta* en el sistema, centrada en torno a cero. El sistema no responde ante estos valores. La magnitud de la zona muerta puede ser modificada, y echa asimétrica por medio de los parámetros del sistema.

Relay: La salida pasa al estado on=1 cuando la entrada supera un valor umbral, y a estado off=0 cuando cae por debajo de un umbral distinto. El estado inicial es off.

Saturation: La señal de salida no sobrepasa un valor umbral, configurable con los parámetros del bloque.

Switch: Una entrada del sistema permite escoger cual de las otras dos entradas se presenta en la salida.

Signals&Systems: Manejo de sistemas y señales.

Subsystem: Permite la realización de sistemas jerárquicos. Al abrir el subsistema, nos permite incluir en su interior, nuevos bloques constructivos, e incluso anidar nuevos subsistemas.

In1: Por defecto un subsistema no contiene entradas. Por cada entrada que se desee añadir se le debe incluir uno de estos bloques.

Out1: Por defecto un subsistema no contiene salidas. Por cada entrada que se desee añadir se le debe incluir uno de estos bloques.

Mux: Permite la inclusión de un conjunto de señales en una única línea de transmisión (que transmite datos vectoriales), lo que facilita la representación en el dibujo. Parámetros: número de entradas. Admite tanto entradas escalares como vectoriales.

Demux: Permite la descomposición de los datos puesto en forma vectorial en una línea mediante un multiplexador. Parámetros: número de salidas.

Data Store Memory: Define una variable del entorno de trabajo que se va a usar como lugar de almacenamiento de datos útil para evitar tener que hacer conexiones complejas que compliquen el diagrama de bloque que se está usando.

Data Store Read: Lee el valor actual de una variable de almacenamiento, que debe estar previamente definida mediante un bloque Data Store Memory

Data Store Write: Cambia el valor actual de una variable de almacenamiento, que debe estar previamente definida mediante un bloque Data Store Memory

Sinks: Sumideros de señales.

Display: Representa numéricamente el valor de una variable.

Scope: Representa gráficamente la evolución en el tiempo de una variable.

To Workspace: Guarda el valor de la señal indicada en una variable del entorno de trabajo del *Matlab*. Se puede escoger el nombre de la misma, y limitar su tamaño.

To File: Guarda en un fichero de tipo ".mat" los datos de la señal de entrada al bloque.

Stop Simulation: Detiene la simulación si el valor de la entrada es distinto de 0.

Sources: Fuentes de señales.

Chirp Signal: Genera una señal senoidal, modulada en frecuencia, entre un valor inicial y un valor final.

Clock: Tiempo que se lleva de simulación.

Constant: Proporciona una señal de valor constante.

From Workspace: Proporciona una secuencia de datos tomadas del entorno de trabajo del *Matlab*. La variable elegida debe contener una matriz indicando los valores de la señal, y los instantes en los que la señal toma estos valores.

From File: Proporciona datos tomados de un fichero ".mat", en el que debe estar el valor de la variable, junto a los instantes de tiempo en que toma cada valor.

Pulse Generator: Genera una onda cuadrada, de la que se puede controlar la amplitud, el periodo y el tiempo de *duty* (relación entre el tiempo que la onda toma su valor máximo y el tiempo que toma el valor mínimo).

Ramp: Genera una señal de tipo *Rampa*.

Random Number: Genera números aleatorios distribuidos según una función normal.

Signal Generator: Simula un generador de señales electrónico, permitiendo generar ondas dientes de sierra, ondas cuadradas y senoidales.

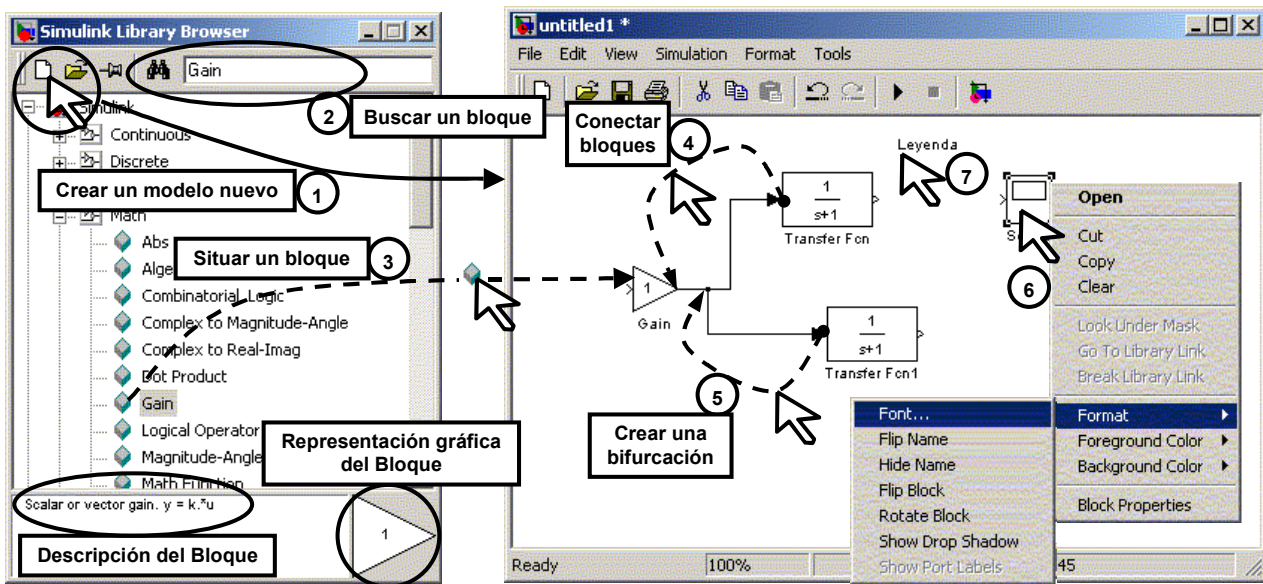
Sine Wave: Generador de ondas senoidales.

Step: Genera una señal de tipo escalón.

Uniform Random Number: Genera números aleatorios distribuidos según una función uniforme.

2. Creación de un modelo.

Para simular un sistema, se deben insertar en las ventanas de simulación los distintos componentes con los que se va a construir el modelo. Se pueden seguir los siguientes pasos:

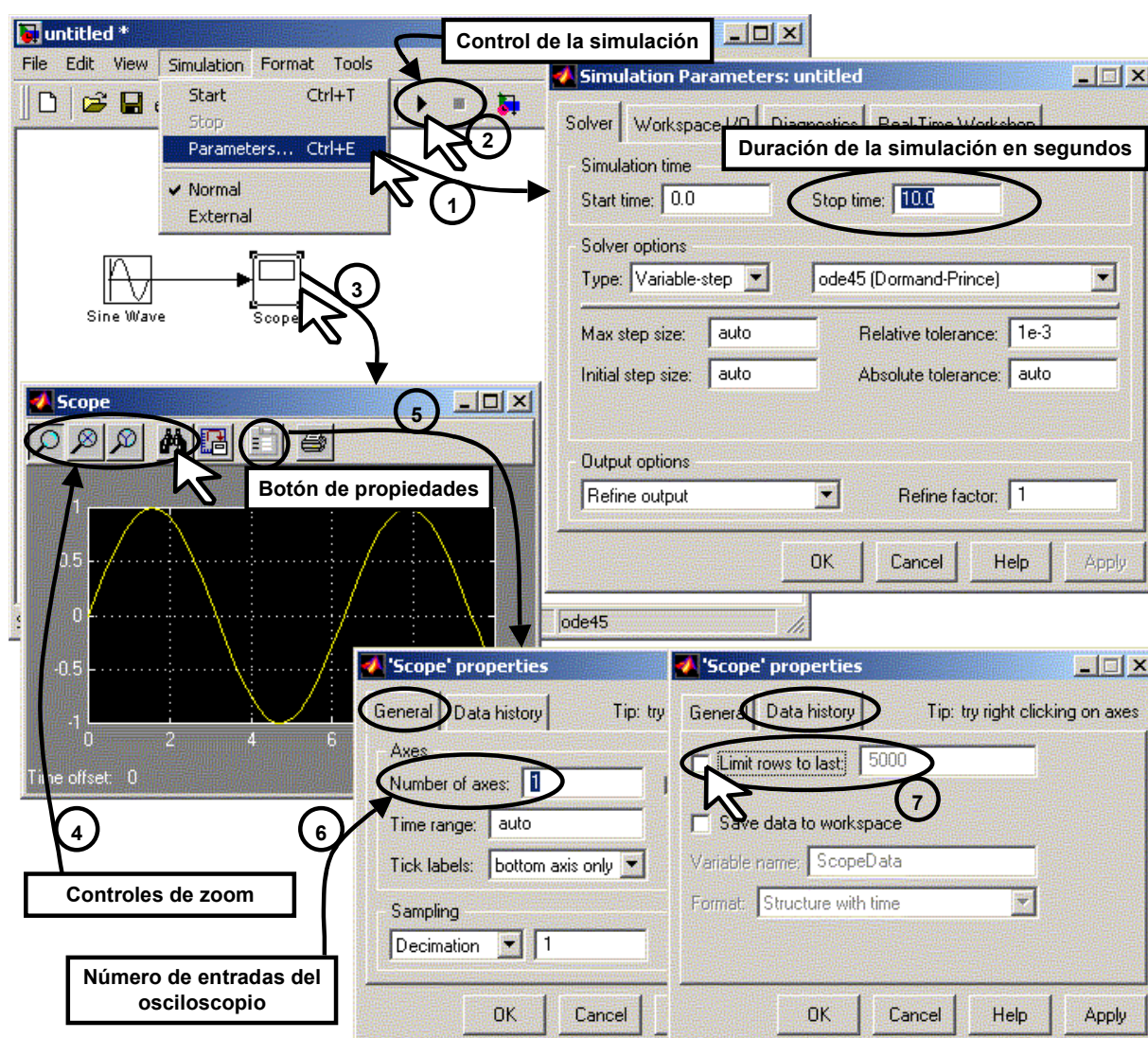


- 1) **Crear un nuevo modelo:** Para abrir una nueva ventana de simulación se debe pulsar el botón "nuevo modelo".
- 2) **Buscar un bloque:** Se puede buscar un bloque expandiendo el árbol de la biblioteca o buscándolo directamente por su nombre en la ventana de búsqueda. En este caso, si hay más de un bloque que pueda corresponder a ese nombre, irán apareciendo a medida que se pulse la tecla "enter" (retorno).
- 3) **Situar un bloque:** Para situar un bloque, se mantiene pulsado el botón izquierdo del ratón sobre el icono en forma de rombo que hay junto al nombre del bloque y se arrastra hacia la posición deseada en la ventana de simulación.
- 4) **Conectar bloques:** En cada bloque, las puntos de salida aparecen indicados mediante una flecha saliente del bloque ">", mientras que los puertos de entrada a cada bloque se indican con una flecha entrante al mismo ">|". Se conecta la entrada de un bloque a la salida de otro, manteniendo pulsado el botón izquierdo del ratón mientras se arrastra desde el símbolo de entrada de uno de los bloques hasta el de salida de otro o viceversa.
- 5) **Crear una bifurcación:** Si se desea llevar la salida de un bloque a la entrada de más de uno se necesita crear una bifurcación en la conexión. Para hacerlo, se arrastra con el ratón desde la entrada del nuevo bloque a conectar hasta la línea de la conexión que se va a bifurcar.
- 6) **Modificar los bloques:** Se pueden rotar o aplicar simetrías a los bloques usados, según convenga la colocación de entradas/salidas para el esquema que se esté realizando, pulsando sobre él el botón derecho del ratón y utilizando los menús desplegables o mediante la opción "Format" del menú principal ("Format/Flip Block", "Format/Rotate Block", etc.). También mediante los menús o haciendo doble clic sobre el bloque, se pueden modificar sus parámetros.
- 7) **Inserción de textos:** Se puede incluir un texto aclaratorio o informativo en cualquier parte de la ventana del modelo, haciendo doble clic en una zona libre y escribiendo directamente el texto.

También se pueden cambiar los nombres y posiciones de los bloques que se empleen para la simulación antes o después de conectarlos. Asimismo los enlaces de las conexiones pueden moverse o modificarse. Para eliminar cualquier elemento basta con seleccionarlo con un clic y eliminarlo con la tecla "sup" o "delete", o utilizar alguno de los menús.

Conviene guardar ("File/Save as") periódicamente el modelo, incluso antes de terminarlo, para evitar perder el trabajo realizado.

Un ejemplo trivial incluiría la selección de dos componentes: "Simulink\Sources\Sine Wave" y "Simulink\Sinks\Scope" de la ventana "Simulink Library Browser", y el arrastre de los mismos hasta la ventana de dibujo. En el caso de nuestro ejemplo básico, para conectar el generador de señales y el osciloscopio, simplemente se debe situar el ratón sobre el punto de salida del generador, pulsar el botón izquierdo, arrastrar el ratón hasta el punto de entrada del osciloscopio y soltar el botón del ratón.



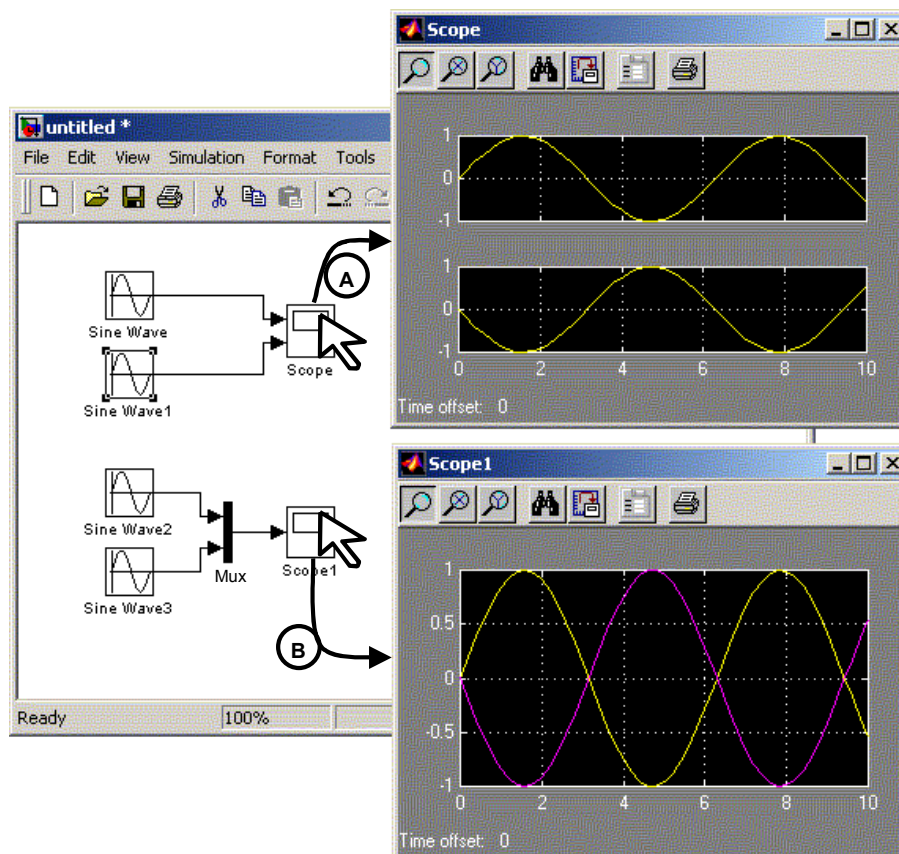
3. Control de la simulación y visualización de resultados.

Antes de poder ejecutar la simulación, es necesario seleccionar los parámetros apropiados para la misma (1). Desde el menú: "Simulation/Parameters", se puede desplegar un cuadro de dialogo, en el que se controlan parámetros de la simulación de entre los cuales el que se modifica

más habitualmente es el tiempo final de la simulación. (Otros parámetros accesibles son el tiempo de inicio de la simulación, el método matemático que se empleará para llevarla a cabo, o las variables que se tomarán/guardarán de/en el espacio de trabajo). La simulación se puede poner en marcha o detener mediante el menú anterior o los botones de la ventana (2).

Para visualizar los resultados de la misma son muy útiles los bloques se encuentran en el grupo "Sinks" de la biblioteca de Simulink. De entre ellos, quizás el más útil es el bloque "Scope" que simula el comportamiento de un osciloscopio. Tras realizar una simulación se pueden ver los resultados que ha registrado haciendo un doble clic sobre él (3). Para ver correctamente los resultados se utilizan los controles de zoom (4), siendo conveniente pulsar siempre tras una simulación el botón de autoescala (el de los prismáticos) para ver el total de los datos registrados. Los otros tres botones de zoom permiten respectivamente ampliar un área señalada con un arrastre del ratón, ampliar el eje "X" de la misma manera o ampliar el eje "Y".

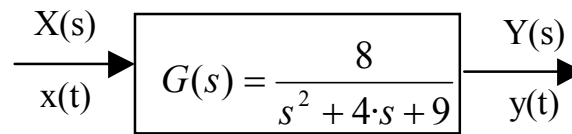
El bloque "Scope" tiene una serie de propiedades a las que se accede a través del botón correspondiente de la ventana "Scope" (5). Dos de las más útiles son la que permite elegir el número de entradas que se desean para el osciloscopio (6) "Number of axes" (que será también el número de gráficos que representará) y la que determina si el osciloscopio almacena todos los datos de la simulación o sólo los últimos obtenidos (7) "Limit rows to last". Respecto a este último control, es conveniente eliminar la marca "✓" del cuadradito blanco para que el osciloscopio mantenga todos los datos registrados durante la simulación completa.



Si se desea visualizar más de una señal en un osciloscopio, existen dos posibilidades:
A) Aumentar el número de entradas del osciloscopio como se comentó anteriormente.
B) Utilizar un bloque "Mux" para que ambas señales aparezcan en el mismo gráfico.

EJEMPLO 1

Simular la respuesta de un sistema descrito por su función de transferencia ante una determinada señal de entrada con Matlab o Simulink es muy sencillo:

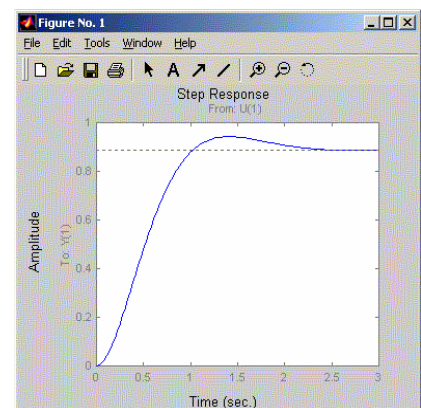
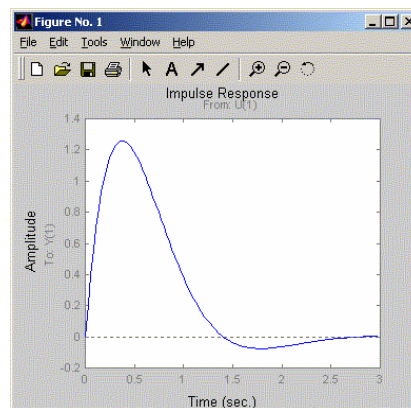


Con Matlab se puede obtener la respuesta ante un impulso de Dirac, un escalón, etc.:

```
>> g = tf([8], [1 4 9])
g=8/(s^2+4s+9)
```

```
>> impulse(g)
```

```
>> step(g)
```



Con Simulink se construye el modelo del sistema, pudiendo optarse por introducir sus parámetros de forma explícita o dejarlos como parámetros a los que se les puede asignar un valor como variables desde la ventana de comandos de Matlab. Una vez ejecutada la simulación se obtendrá en el bloque “Scope” el resultado de la misma.

EJEMPLO 2

Si se desea simular un sistema más complejo basta con trasladar las ecuaciones de su modelo a un modelo de bloques de Simulink.

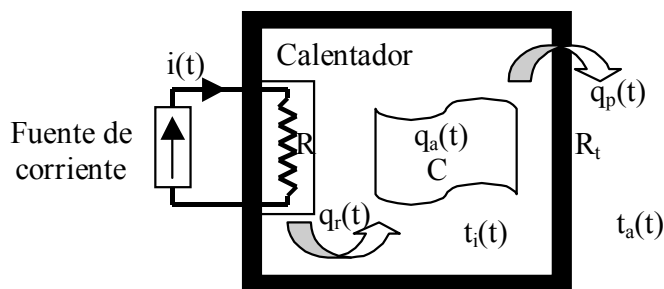
Modelo del sistema:

$$q_r(t) = q_a(t) + q_p(t)$$

$$q_r(t) = R \cdot i^2(t)$$

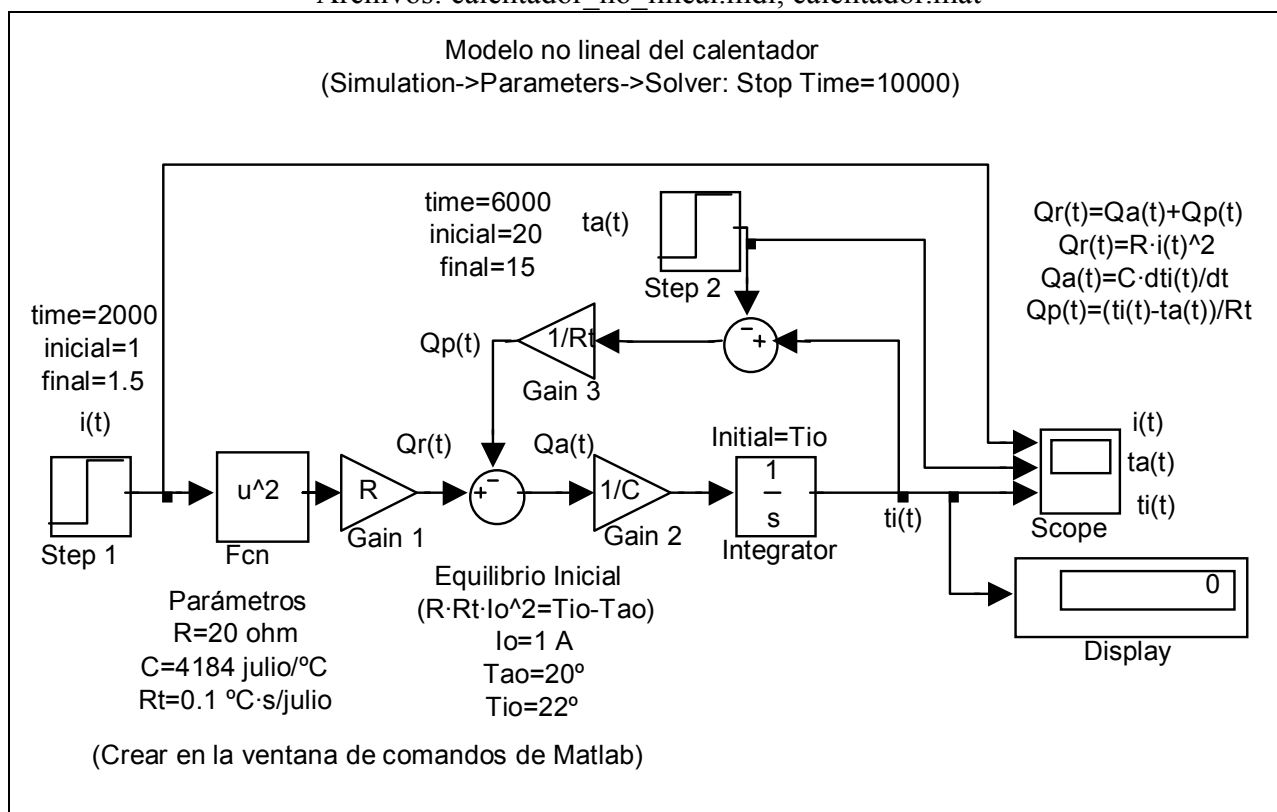
$$q_a(t) = C \cdot dt_i(t)/dt$$

$$q_p(t) = (t_i(t) - t_a(t))/R_t$$



Simulink puede simular sistemas no lineales, como se puede ver en el siguiente modelo, siendo necesario en muchos casos definir los valores iniciales de algunas de las variables del sistema (en bloques como los “integrator”). Los valores del modelo representado que se han dejado como parámetros son asignados en la ventana de comandos de Matlab (debe tenerse cuidado con el uso de mayúsculas y minúsculas en la denominación de estos parámetros). Los archivos de ejemplo indicados contienen el modelo representado y el espacio de trabajo con los valores asignados a los parámetros para la ventana de comandos de Matlab.

Archivos: calentador_no_lineal.mdl, calentador.mat



También se puede linealizar el modelo respecto a un punto de funcionamiento y construir el modelo en transformadas de Laplace. Luego se trasladan las ecuaciones del modelo linealizado a un modelo de Simulink mediante, por ejemplo, bloques “Función de Transferencia” (TransferFcn).

Modelo linealizado del sistema en Transformadas de Laplace:

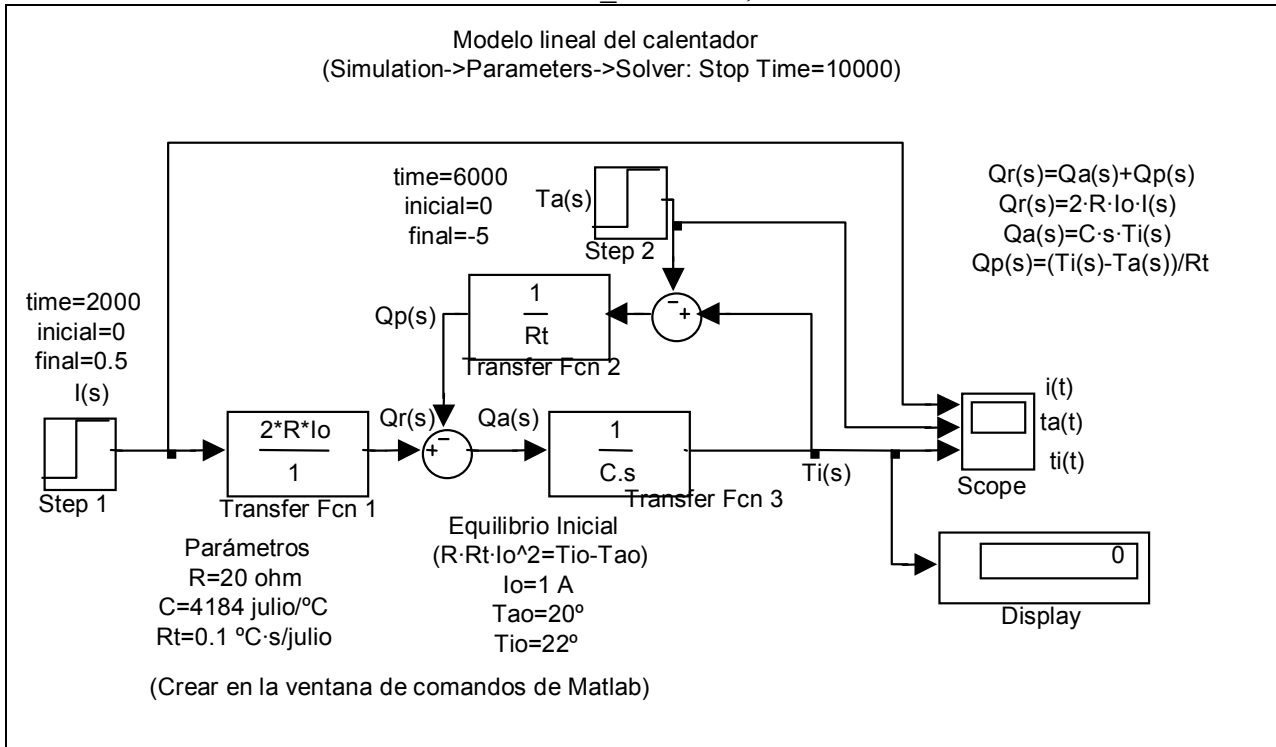
$$Q_r(s) = Q_a(s) + Q_p(s)$$

$$Q_r(s) = 2 \cdot R \cdot I_o \cdot I(s)$$

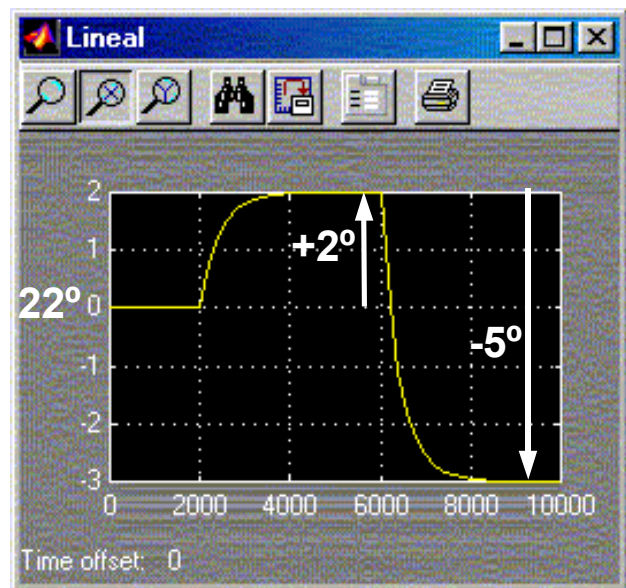
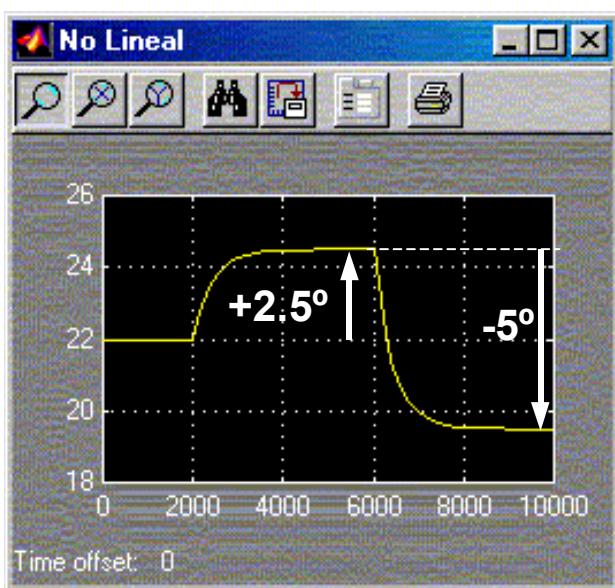
$$Q_a(s) = C \cdot s \cdot T_i(s)$$

$$Q_p(s) = (T_i(s) - T_a(s)) / R_t$$

Archivos: calentador_lineal.mdl, calentador.mat

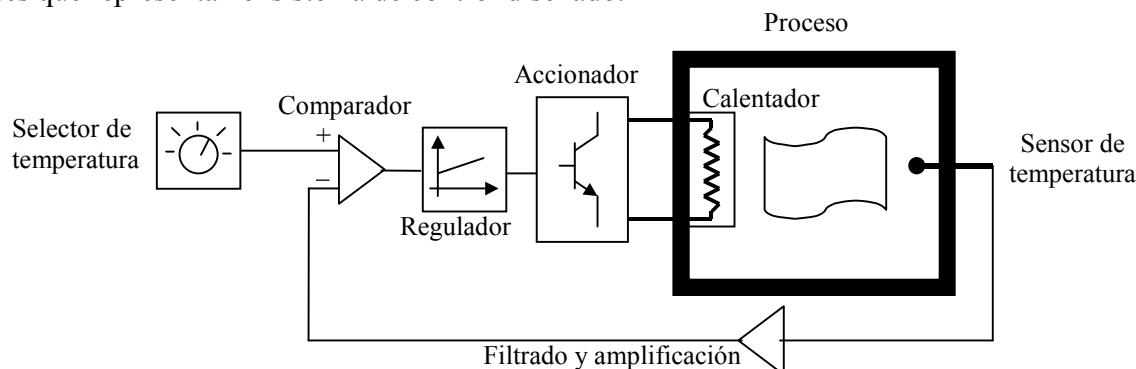


Al comparar los resultados de la simulación de ambos modelos, se puede observar el error que introduce la linealización del modelo.

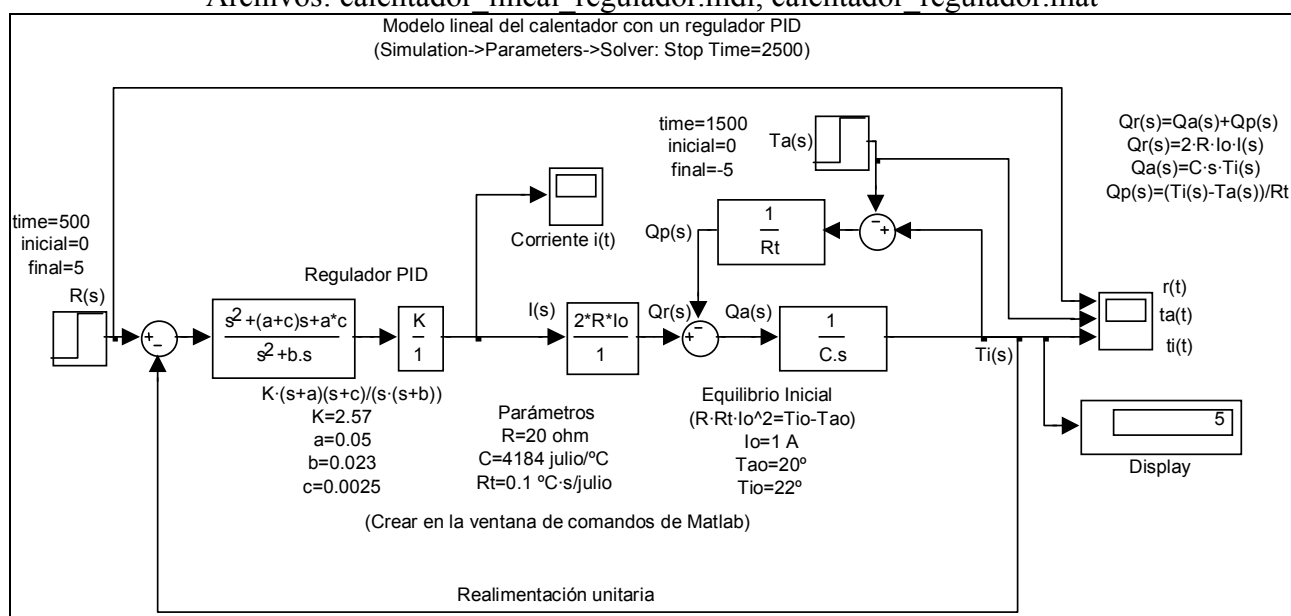


EJEMPLO 3

Una vez diseñado un regulador para un sistema, basándose en su modelo linealizado, se puede comprobar con Simulink si el comportamiento final del sistema es adecuado. Basta con añadir los bloques que representan el sistema de control diseñado.

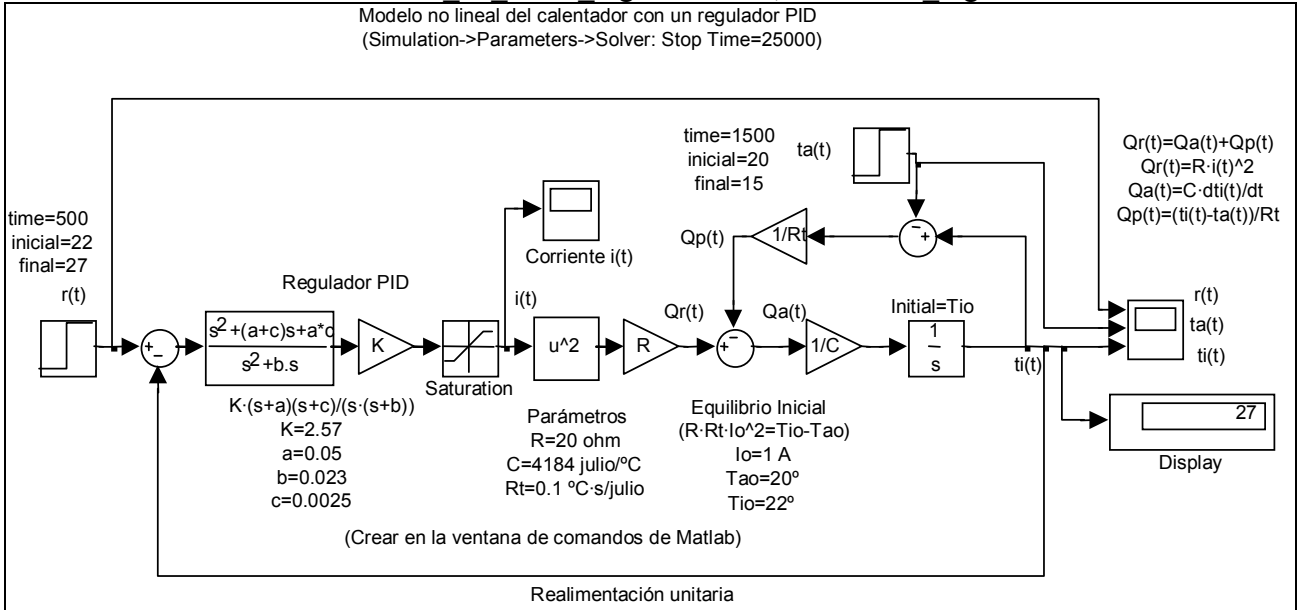


Archivos: calentador_lineal_regulador.mdl, calentador_regulador.mat

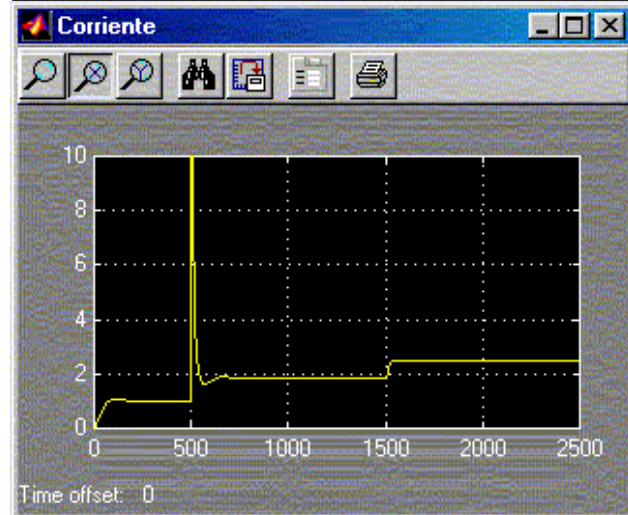
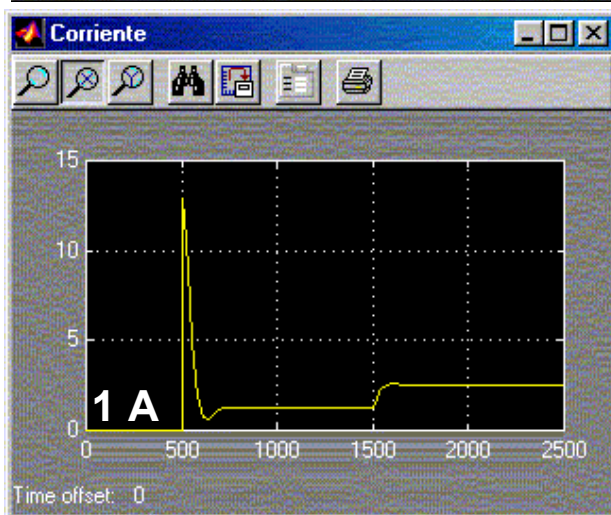
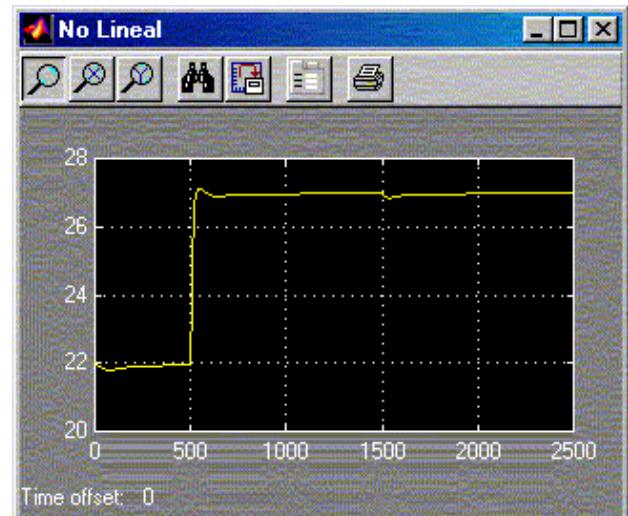
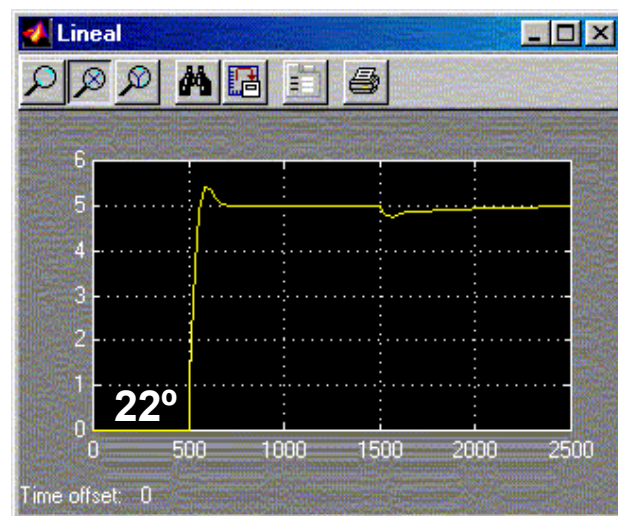


Sin embargo, siempre que sea posible, es conveniente comprobar el comportamiento del sistema de control sobre el modelo no lineal del sistema. Pueden tenerse en cuenta, por ejemplo, limitaciones en la evolución de determinadas variables del sistema. En el siguiente ejemplo, a parte de utilizar el modelo no lineal que se tenía para el sistema, se ha supuesto que la corriente máxima $i(t)$ que se puede conseguir es de 10 A y que su valor nunca es negativo. Esto se ha reflejado en el modelo mediante un bloque denominado "Saturation".

Archivos: calentador_no_lineal_regulador.mdl, calentador_regulador.mat



Los resultados muestran (aunque no son exactamente iguales en una y otra simulación) que el sistema de control sigue comportándose correctamente.





EJERCICIO 1: Visualización de la respuesta de un sistema

Construir una función de transferencia sustituyendo los dígitos ABCDEFGH con los números de su D.N.I., ajustados a la derecha y rellenando con un cero por la izquierda si es necesario:

DNI : . .
 -- --- ---
 AB CDE FGH

$$G(s) = \frac{1AB*s+2CD}{s^3+1E*s^2+5F*s+1GH}$$

Por ejemplo:

DNI : **09.345.678**
 -- --- ---
 AB CDE FGH

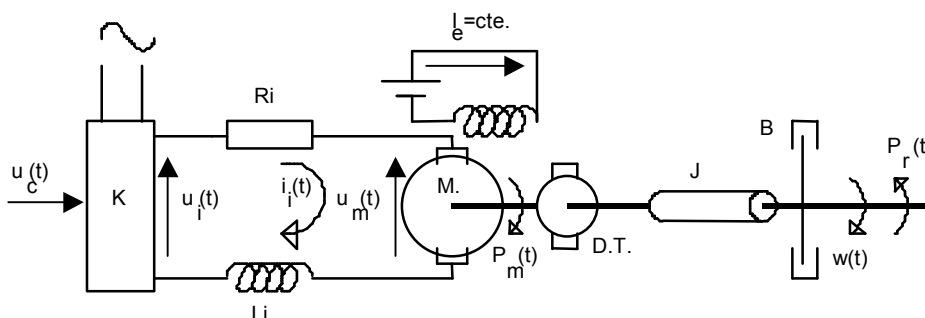
$$G(s) = \frac{109*s+234}{s^3+15*s^2+56*s+178}$$

Incluir en el informe los siguientes datos y resultados:

(Indicar **todos los comandos y el diagrama de bloques del modelo en Simulink utilizados** para obtener los datos que se solicitan, **los resultados numéricos** correspondientes, y los **dibujos a mano alzada de los gráficos solicitados**)

1. Apellidos, Nombre y DNI del alumno.
2. La función de Transferencia G(s) creada.
3. Los ceros y polos del sistema G(s).
4. Respuesta ante un impulso. (con Matlab)
 - 4.1. Dibujo de la respuesta.
 - 4.2. Máximo de la respuesta.
 - 4.3. Valor en Régimen permanente.
5. Respuesta ante un escalón. (con Simulink)
 - 5.1. Dibujo de la respuesta.
 - 5.2. Máximo de la respuesta.
 - 5.3. Valor en Régimen permanente.

EJERCICIO 2: Creación de un modelo de simulación



El sistema de la figura está compuesto por los siguientes elementos:

- Un rectificador controlado que alimenta a un motor CC con una tensión continua $u_i(t)$ proporcional a la tensión $u_c(t)$, con constante de proporcionalidad K .
- Un motor CC con corriente de excitación constante de parámetros: R_i , L_i , cte. contraelectromotriz K_b y cte. de par del motor K_p .
- El conjunto rotor-eje del motor y de la D.T. tiene una inercia J y un coeficiente de rozamiento viscoso B (que se representan en la figura). En el extremo del eje existe un par resistente variable, $p_r(t)$, debido a los elementos mecánicos que mueve el motor y que no aparecen en la figura.

Modelo matemático del sistema:

$$\begin{aligned} u_i(t) &= K \cdot u_c(t) \\ u_i(t) &= R_i \cdot i_i(t) + L_i \cdot di_i(t)/dt + u_m(t) \\ u_m(t) &= K_b \cdot w(t) \\ p_m(t) &= K_p \cdot i_i(t) \\ p_m(t) &= p_r(t) + J \cdot dw(t)/dt + B \cdot w(t) \end{aligned}$$

En transformadas de Laplace:

$$\begin{aligned} U_i(s) &= K \cdot U_c(s) \\ U_i(s) &= R_i \cdot I_i(s) + L_i \cdot s \cdot I_i(s) + U_m(s) \\ U_m(s) &= K_b \cdot W(s) \\ P_m(s) &= K_p \cdot I_i(s) \\ P_m(s) &= P_r(s) + J \cdot s \cdot W(s) + B \cdot W(s) \end{aligned}$$

Obtenga el valor de los parámetros de las ecuaciones sustituyendo convenientemente los dígitos de su D.N.I. en las expresiones siguientes:

D.N.I.: $\frac{A}{A} \frac{B}{B} \cdot \frac{C}{C} \frac{D}{D} \frac{E}{E} \cdot \frac{F}{F} \frac{G}{G} \frac{H}{H}$

$$\begin{aligned} K &= 1 + A \\ K_b &= 0.2 - 0.01 \cdot B \quad [\text{V} \cdot \text{s}/\text{rad}] \\ K_p &= 1 + 0.2 \cdot C \quad [\text{N} \cdot \text{m}/\text{A}] \\ R_i &= 10 - D \quad [\Omega] \\ L_i &= 0.01 + 0.02 \cdot (E + F) \quad [\text{H}] \\ J &= 10 + G \quad [\text{Kg} \cdot \text{m}^2] \\ B &= 5 - (0.1 \cdot H) \quad [\text{Kg} \cdot \text{m}^2/\text{s}] \end{aligned}$$

Por ejemplo:

D.N.I.: $\frac{0}{A} \frac{9}{B} \cdot \frac{3}{C} \frac{4}{D} \frac{5}{E} \cdot \frac{6}{F} \frac{7}{G} \frac{8}{H}$

$$\begin{aligned} K &= 1 + 0 = 1 \\ K_b &= 0.2 - 0.01 \cdot 9 = 0.11 \quad [\text{V} \cdot \text{s}/\text{rad}] \\ K_p &= 1 + 0.2 \cdot 3 = 1.6 \quad [\text{N} \cdot \text{m}/\text{A}] \\ R_i &= 10 - 4 = 6 \quad [\Omega] \\ L_i &= 0.01 + 0.02 \cdot (5 + 6) = 0.23 \quad [\text{H}] \\ J &= 10 + 7 = 17 \quad [\text{Kg} \cdot \text{m}^2] \\ B &= 5 - (0.1 \cdot 8) = 4.2 \quad [\text{Kg} \cdot \text{m}^2/\text{s}] \end{aligned}$$

Construir el modelo del sistema en Simulink considerando a $u_c(t)$ como entrada, $p_r(t)$ como perturbación y $w(t)$ como salida.



Incluir en el informe los siguientes datos y resultados:

(Dibujar **el diagrama de bloques del modelo en Simulink utilizado** para obtener los datos que se solicitan, **los resultados numéricos** correspondientes, y los **dibujos a mano alzada de los gráficos solicitados**)

1. Apellidos, Nombre y DNI del alumno.
2. Dibujo del diagrama de bloques usado en las simulaciones.
3. Valores de las Constantes utilizadas (K , K_p , L_i , ...).
4. Simular la respuesta del sistema a un escalón de cinco unidades en la entrada $U_c(t)$.
 - 4.1. Obtener el valor al que tiende la respuesta en régimen permanente.
 - 4.2. Determinar la constante de tiempo T del sistema.
5. Añadir a la simulación el efecto de un escalón de cinco unidades en la perturbación, una vez que la respuesta al escalón del apartado 4 ha alcanzado el régimen permanente.
 - 5.1. Dibujar aproximadamente la respuesta.
 - 5.2. Obtener el valor al que tiende la respuesta en régimen permanente.
6. Simular la respuesta del sistema ante una entrada senoidal en $U_c(t)$ de pulsación $\omega=1$ rad/s y amplitud 5. (Recuerde anular la perturbación introducida en el apartado 5)
 - 6.1. Dibujar las senoides en régimen permanente.
 - 6.2. Obtener el valor de la amplitud de la senoide de la salida una vez alcanzado el régimen permanente.
 - 6.3. Obtener la diferencia de fase entre la entrada y la salida una vez alcanzado el régimen permanente.

EJERCICIO 3: Estudio de la respuesta en frecuencia de un sistema y bucle cerrado

Construir una función de transferencia sustituyendo los dígitos ABCDEFGH con los números de su D.N.I., ajustados a la derecha y rellenando con un cero por la izquierda si es necesario:

DNI : . .
 -- -- --
 AB CDE FGH

$$G(s) = \frac{1\underline{AB}s + 2\underline{CD}}{s^3 + 1\underline{E}s^2 + 5\underline{F}s + 1\underline{GH}}$$

Por ejemplo:

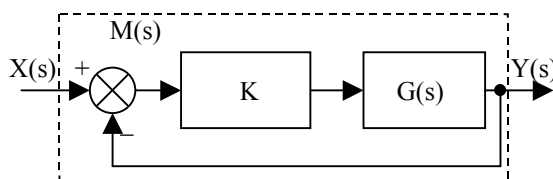
DNI : **09.345.678**
 -- -- --
 AB CDE FGH

$$G(s) = \frac{109s + 234}{s^3 + 15s^2 + 56s + 178}$$

Incluir en el informe los siguientes datos y resultados:

(Indicar **todos los comandos utilizados** para obtener los datos que se solicitan, **los resultados numéricos** correspondientes, y los **dibujos a mano alzada de los gráficos solicitados**)

1. Apellidos, Nombre y DNI del alumno.
2. Indicar la función de Transferencia $G(s)$ creada.
3. Obtener ceros y polos del sistema $G(s)$.
4. Dibujar el diagrama de Bode y obtener el valor del módulo (en decibelios) y la fase (en grados) de la respuesta en frecuencia para $\omega=5$ rad/s de $G(s)$.
5. Obtener el valor del módulo (en decibelios) en el pico de resonancia y la frecuencia de resonancia de $G(s)$. (!ATENCIÓN! Puede que el pico de resonancia no exista o no coincida con el valor máximo en módulo del diagrama de Bode. Indicar si suceden estas situaciones).
6. ¿Cuánto vale “ M_r ”?
7. Obtener la función de transferencia $M(s)$ resultado de realimentar unitaria y negativamente a $G(s)$ con $K=1$.

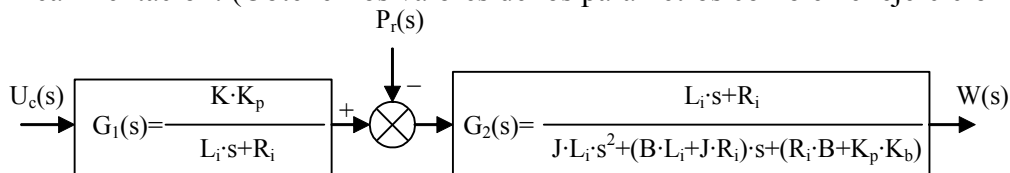


8. Obtener los polos del sistema en bucle cerrado $M(s)$ para $K= 5$.
9. Obtener los márgenes de ganancia y de fase y las frecuencias correspondientes a esos márgenes para $G(s)$ (En algunos casos pueden ser infinito).

EJERCICIO 4: Diseño de un regulador para un modelo de simulación

Se pretende diseñar un regulador para el sistema del ejercicio 2 cuando este se realimenta negativamente mediante una tacodinamo. La tacodinamo proporciona una tensión de salida de un voltio por cada 1.000 r.p.m. con una respuesta dinámica semejante a un sistema de primer orden con constante de tiempo $T=0.1$ segundos.

Sistema sin realimentación: (Obtener los valores de los parámetros como en el ejercicio 2)



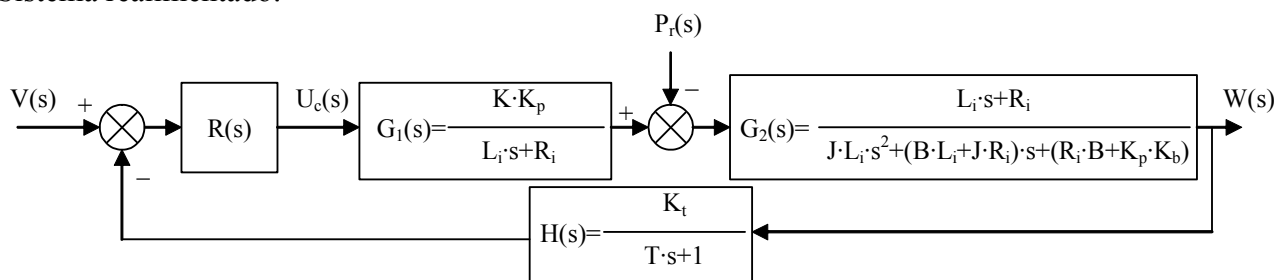
Especificaciones para el regulador:

Tiempo de establecimiento: $t_s \leq 0.4$ segundos Sobreoscilación: $M_p \leq 5\%$

Anular el error de posición, e_p , en régimen permanente.

Eliminar el efecto de perturbaciones en el par $p_i(t)$ sobre el régimen permanente del sistema.

Sistema realimentado:



Incluir en el informe los siguientes datos y resultados:

(Dibujar el **diagrama de bloques del modelo en Simulink utilizado** para obtener los datos que se solicitan, **los resultados numéricos** correspondientes, y los **dibujos a mano alzada de los gráficos solicitados**)

1. Apellidos, Nombre y DNI del alumno.
2. Valores de las Constantes utilizadas (K , K_p , L_i , ...) incluyendo las de la tacodinamo.
3. Regulador diseñado para el control incluyendo los cálculos realizados para obtenerlo. Utilizar a ser posible el criterio de la vertical para el diseño: si el ángulo de compensación obtenido para el regulador sale negativo cambiar la especificación de t_s por $t_s \leq 0.2$ segundos y si sale mayor de 90° cambiar por $t_s \leq 1$. Si persisten los problemas consulte con el tutor.
4. Dibujo del diagrama de bloques en bucle cerrado que incluye el regulador utilizado con Simulink.
5. Simular la respuesta del sistema a un escalón de cinco unidades en la entrada $v(t)$.
 - 5.1. Obtener aproximadamente el valor de la sobreoscilación M_p .
 - 5.2. Obtener aproximadamente el valor del tiempo de pico t_p .
 - 5.3. Obtener el valor al que tiende la respuesta del sistema, $w(t)$, en régimen permanente.
6. Simular el efecto de un escalón de 2.000 unidades en la perturbación $p_i(t)$ una vez que la respuesta anterior ha alcanzado el régimen permanente y dibujar la respuesta del sistema.