PRACTICA 0.
Fundamentos de Matlab para el estudio y análisis de sistemas de control

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

0. Introducción al software de simulación MATLAB.

Por su versatilidad y su uso extendido el paquete de software MATLAB se presenta como una herramienta potente para la resolución de problemas de ingeniería de control.

En esta práctica se va a presentar diferentes tipos de órdenes y funciones de las que dispone MATLAB y que van a ser utilizadas frecuentemente en las diferentes metodologías y técnicas que se usan para abordar el control de sistemas físicos, así como para simular el comportamiento de éstos.

0.1 Iniciar Matlab.

Una vez que se ha ejecutado el software MATLAB desde el sistema operativo empleado haciendo click con el ratón sobre el icono correspondiente, apareceré la ventana de comandos que se identifica por el indicador de ejecución de comandos `>>`. Este indicador avisa de que el MATLAB esta preparado para recibir la ejecución algún comando o rutina.

El MATLAB dispone de un comando que muestra una ayuda sobre los diferentes comandos disponibles. Para ejecutar este comando basta teclear la palabra `help` después del indicador de ejecución y pulsar la tecla [INTRO]:

```
>> help
```

0.2 Variables de Matlab.

El MATLAB automáticamente crea siempre una variable llamada `ans` que guarda el resultado de la última operación cuando no se indica ninguna variable donde guardar dicho resultado.

Si se ejecuta:

```
>> a=2
>> b=3
>> a*b
>> ans
```

uno se encuentra que `ans` guarda el resultado de `a*b`.

```
ans =
     6
```

MATLAB también distingue entre mayúsculas y minúsculas y para ejecutar funciones y rutinas predefinidas deben escribirse en minúsculas.

0.3 Salir de Matlab.

La forma más sencilla de salir de MATLAB es ejecutar el comando `quit` o `exit` en la ventana de comandos. Al terminar una sesión de MATLAB las variables en el espacio de trabajo se borran. Para guardar el espacio de trabajo, de modo que las variables definidas se mantengan en próximas sesiones se ejecuta el comando `save`. El comando `save` guarda todas las variables definidas en la sesión en un archivo llamado `matlab.mat`

Si lo que se desea es guardar solo algunas variables se puede utilizar parámetros con el comando `save` indicando el nombre de las variables y el nombre del fichero en el que se quieren guardar. Por ejemplo, la siguiente sentencia almacena en el fichero `misvariables.mat` el contenido de las variables `A B` y `C`.

```
>> save misvariables.mat A B C
```

Por otro lado, si lo que se desea es volver a cargar las variables previamente guardadas bastaría ejecutar uno de los siguientes comandos, en función de si lo que se desea es cargar el fichero `matlab.mat` o el fichero `misvariables.mat`.

```
>> load
>> load misvariables
```

Si lo que se quiere es listar las variables disponibles en el entorno de trabajo, bastará con ejecutar el comando `who`, o el comando `whos` si lo que se desea es ver información adicional sobre dichas variables.

1. Introducción de datos.

1.1 Vectores.

El modo más sencillo de introducir una secuencia de datos en MATLAB es hacerlo mediante una lista explícita de elementos, donde los elementos deben estar separados por espacios en blanco o por comas, y se encuentran encerrados entre corchetes:

```
>> x=[1 2 3 4 5]
     ó
>> x=[1,2,3,4,5]

x =
     1     2     3     4     5
```

De ahí que el comando anterior creará un vector fila que contiene los elementos 1, 2, 3, 4 y 5. Si lo que queremos es introducir los datos como un vector columna habría que haber puesto:

```
>> x=[1 2 3 4 5]'
```

```
     ó
>> x=[1,2,3,4,5]'


x =
     1
     2
     3
     4
     5
```

1.2 Matrices.

Para representar una matriz hay que tener en cuenta 3 reglas básicas, que son:

- Los elementos deben estar separados por comas o blancos.
- Los elementos deben estar encerrados entre corchetes.
- El final de cada fila se indica con un punto y coma (;) o mediante un retorno de carro.

Así, se puede representar una matriz como sigue:

```
>> A=[1 2 3; 4 5 6; 7 8 9]
     ó

>> A=[1 2 3
      4 5 6
      7 8 9]

A =
     1     2     3
     4     5     6
     7     8     9
```

Los elementos de una matriz pueden ser cualquier expresión matemática evaluable por MATLAB. Un ejemplo podría ser:

```
>> B=[1 exp(-0.08); sqrt(2) 2]

B =
  1.0000    0.9231
  1.4142    2.0000
```

Cómo ya se vio en el apartado anterior, el apostrofe (') indica la traspuesta conjugada de una matriz. Si la matriz es real, la traspuesta conjugada consiste en transponer los elementos de la matriz, es decir cambiar filas por columnas.

```
>> C=A'
```

dará como resultado que la matriz C es la matriz traspuesta de la matriz A.

```
C =
```

```
     1     4     7
     2     5     8
     3     6     9
```

1.3 Números complejos.

Para representar números complejos con MATLAB se utiliza la variable i o j. Por ejemplo, si se quiere representar el número $1+\sqrt{2}\,j$ se procede de la siguiente manera:

```
>> x= 1+sqrt(2)*i
      ó
>>  x= 1+sqrt(2)*j

x =
   1.0000 + 1.4142i
```

Por ejemplo, el número complejo 2+3*j se debe introducir como:

```
>> x=2+3*j

x =
   2.0000 + 3.0000i
```

Si i y j se utilizan como variables, habría que renombrar la variable compleja como `ii=sqrt(-1)` ó `jj=sqrt(-1)` y entonces el número 2+3*j se debería introducir como:

```
>> jj=sqrt(-1)
>> x=2+3*jj

jj =
   0 + 1.0000i
x =
   2.0000 + 3.0000i
```

1.4 Matrices complejas.

Una matriz compleja no es más que una representación matricial como la vista en el apartado 1.2 en la que los elementos que la constituyen son números complejos como los vistos en el apartado 1.4. Teniendo en cuenta esto, un ejemplo de matriz compleja podría ser:

```
>> X=[1 j; 2+3*j 2]

X =

   1.0000                  0 + 1.0000i
   2.0000 + 3.0000i   2.0000
```

Es importante destacar que, cuando se introducen números complejos como elementos de matrices entre corchetes, se evitan los espacios en blanco.

Si queremos obtener la traspuesta no conjugada de esta matriz bastaría con hacer `Y=X'`. Si por el contrario queremos obtener la traspuesta conjugada se pondría `Y=X.'` o `Y=conj(X')`.

2. Operaciones con datos.

2.1 Suma y resta de matrices.

Para sumar o restar con matrices hay que tener en cuenta que cada una de las matrices operando deben tener las mismas dimensiones. Dadas dos matrices A y B.

```
>> A=[1 2 3; 4 5 6; 7 8 9]
>> B=[9 8 7; 6 5 4; 3 2 1]
```

Se puede obtener una matriz C que resulta de sumar A y B como:

```
>> C=A+B

C =
    30    24    18
    84    69    54
   138   114    90
```

Y la matriz D que resulta de restar A y B como:

```
>> D=A-B

D =
    -8    -6    -4
    -2     0     2
     4     6     8
```

2.2 Multiplicación de matrices.

Para multiplicar matrices hay que tener en cuenta que las columnas de una de las matrices coinciden con el número de filas de la otra matriz. Dadas dos matrices A y X, donde el número de columnas de A es igual al número de filas de X.

```
>> X=[ 1; 2; 3]
>> A=[9 8 7; 6 5 4; 3 2 1]
```

Se puede obtener una matriz Y que resulta de multiplicar A por X como:

```
>>Y=A*X
```

```
Y =
    46
    28
    10
```

2.3 Matriz exponencial.

La matriz exponencial de una matriz dada A de dimensiones *nxn* viene dada por el desarrollo $I + A + A^2/2! + A^3/3! + \ldots$ y puede ser calculada en MATLAB con el comando `expm(A)`.

Así dada una matriz A como la del apartado 2.2, se puede obtener su exponencial como:

```
>> B=expm(A)

B=
  1.0e+006 *

    5.7552    4.8520    3.9489
    3.6929    3.1134    2.5339
    1.6307    1.3748    1.1189
```

2.4 Valor absoluto de una matriz.

La matriz valor absoluto de una matriz dada A, corresponde con una matriz cuyos elementos son el valor absoluto de cada uno de los elementos de A. Y se calcula como:

```
>> B=abs(A)
```

Si la matriz A es una matriz compleja, `abs(A)` devuelve el módulo del complejo. Es decir:

```
>> A=[1 j; 2+3*j 2]

A =
   1.0000                  0 + 1.0000i
   2.0000 + 3.0000i   2.0000

>> real(A)

ans =
     1     0
     2     2


>> imag(A)

ans =
     0     1
     3     0

>> sqrt(real(A).^2+imag(A).^2)
```

```
ans =
    1.0000    1.0000
    3.6056    2.0000
```

Además con la función `angle(A)` se obtienen los ángulos de fase en radianes de los elementos de dicha matriz compleja. Esos ángulos estarán comprendidos entre $[-\pi,\pi]$. Para comprender esto, veamos un ejemplo.

Se quiere calcular el módulo y la fase de los elementos de la matriz A. Si la matriz A se introduce como:

```
>> A=[2+2*i 1+3*i;4+5*i 6-i]

A =
   2.0000 + 2.0000i   1.0000 + 3.0000i
   4.0000 + 5.0000i   6.0000 - 1.0000i
```

El modulo de dicha matriz se obtendrá como:

```
>> M=abs(A)

M =
    2.8284    3.1623
    6.4031    6.0828
```

La fase de A se obtendrá como:

```
>> F=angle(A)

F =
    0.7854    1.2490
    0.8961   -0.1651
```

2.5 Magnitud y fase de un número complejo.

Al igual que con una matriz, si se quiere obtener el módulo y la fase de un número complejo $z = x + yj = r \cdot e^{j\vartheta}$, se procede del siguiente modo:

```
r= abs(z)
theta= angle(z)
```

donde la sentencia z=r*exp(j*teta) vuelve a obtener el número complejo en su forma normal.

```
>> z=3+i*2
z =
  3.0000 + 2.0000i

>>r=abs(z)
```

```
r =
   3.6056

>> theta=angle(z)
theta =
    0.5880
```

2.6 Otras operaciones con matrices y vectores.

Si se desea obtener una matriz B donde cada uno de los elementos sea el cuadrado de los elementos de una matriz A se utilizaría la instrucción siguiente:

```
B=A.^2
```

Si se desea multiplicar, uno a uno, los elementos correspondientes de dos matrices A y B (que ocupan la misma posición de fila y columna) se ejecutaría la siguiente instrucción:

```
C=A.*B
```

Si se desea dividir, uno a uno, los elementos correspondientes de dos matrices A y B (que ocupan la misma posición de fila y columna) se ejecutaría la siguiente instrucción:

```
C=A./B
```

Si lo que se quiere es dividir los elementos B con sus correspondientes en A, entonces podríamos poner:

`C=B./A` ó `C=A.\B`

2.7 Matrices especiales.

La matriz identidad *I* es una matriz cuadrada de dimensiones *nxn* cuyos elementos son cero a excepción de su diagonal cuyos valores son 1. Para definir una matriz identidad en MATLAB se usa la sentencia `eye(n)`, donde *n* indica las dimensiones de la matriz:

```
>> I=eye(5)
I =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

dará una matriz 5x5 con unos en la diagonal y el resto ceros.

Una matriz diagonal es una matriz cuadrada cuyos elementos son cero a excepción de su diagonal cuyos valores son un número determinado *m*. Si *x* es un vector, la orden `diag(x)` produce una matriz diagonal con *x* sobre la diagonal. Por ejemplo:

```
>> x=[1 2 3]
>> D=diag(x)
```

```
D =
     1     0     0
     0     2     0
     0     0     3
```

dará una matriz 3x3 con 1, 2, y 3 en la diagonal y el resto ceros.

En la tabla 1 se muestran otras funciones:

Nombre de la función	Cometario
`ones(n)`	Crea una matriz nxn de unos.
`ones(m,n)`	Crea una matriz mxn de unos.
`zeros(n)`	Crea una matriz nxn de ceros
`zeros(m,n)`	Crea una matriz mxn de ceros
`diag(A)`	Obtiene la diagonal de una matriz A

***Tabla 1**. Funciones de matrices especiales.*

3. Representación gráfica de curvas.

MATLAB dispone de un conjunto de rutinas para obtener diferentes tipos de salidas gráficas. El modo de utilización de cada una de ellas es muy similar, siendo la única diferencia entre todas ellas el modo en el que se escalan los ejes y el modo de visualización los datos.

Las rutinas básicas para construir gráficas se muestran en la siguiente tabla:

Nombre de la función	Comentario
`plot`	Crea un gráfico a partir de dos vectores de datos o columnas de matrices
`subplot`	Permite crear múltiples gráficos en una misma figura
`loglog`	Crea un gráfico con escala logarítmica para ambos ejes
`semilogx`	Crea un gráfico con escala logarítmica para el eje x, y escala lineal para el eje y
`semilogy`	Crea un gráfico con escala logarítmica para el eje y, y escala lineal para el eje x
`title`	Añade título al gráfico
`xlabel`	Añade título al eje x
`ylabel`	Añade título al eje y
`text`	Añade texto en una localización determinada
`gtext`	Añade texto a la gráfica
`grid`	Crea líneas de rejilla
`polar(theta,ro)`	Crea un gráfico en coordenadas polares de ángulo theta frente a radio ro.

***Tabla 2**. Funciones de construcción y manejo de gráficas.*

3.1 Representación gráfica de curvas.

Para dibujar una gráfica a partir de dos vectores x e y de la misma longitud se procede de la siguiente manera: `plot(x,y)` dibuja los valores de y frente a los valores de x. Por ejemplo:

```
>> x=[1 2 3 4 5 6 7]
>> y=[2 4 9 16 25 36 49]
>>plot(x,y)
```

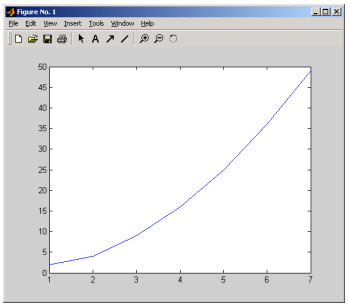


***Figura 1**. Ejemplo de construcción de una gráfica sencilla.*

3.2 Representación de varias curvas en una misma gráfica.

Para dibujar varias curvas en una misma gráfica se utiliza la orden `plot` con múltiples argumentos. Por ejemplo si se quiere dibujar n curvas cuyos valores se encuentran en los vectores x e y de cada una de ellas, la rutina sería:

```
plot(x1,y1,x2,y2,x3,y3,...,xn,yn)
```

donde x1 e y1 son los vectores de los valores de la primera curva, x2 e y2 los vectores de los valores de la segunda curva y así sucesivamente hasta n curvas.

Por ejemplo:

```
>> x=[1 2 3 4 5 6 7]
>> y=[1 4 9 16 25 36 49]
>> z=[1 8 27 64 125 216 343]
>>plot(x,y,x,z)
```

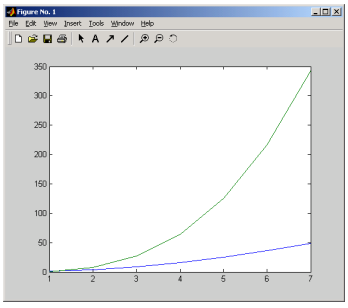


***Figura 2**. Ejemplo de construcción de varias gráficas en una misma ventana.*

Otra forma de dibujar más de una curva en un único gráfico se consigue utilizando la orden `hold`. Esta orden congela el gráfico actual e inhibe las acciones de borrado y escalado. De modo, que usando esta orden, las curvas que se dibujen después de la ejecución de este comando se dibujarán sobre la curva original, superponiendo unas sobre otras. Si se desea

activar de nuevo el redibujado, permitiendo el borrado y escalado, será necesario volver a ejecutar la orden `hold`.

```
>> x=[1 2 3 4 5 6 7]
>> y=[1 4 9 16 25 36 49]
>> plot(x,y)
>> hold
>> z=[1 8 27 64 125 216 343]
>> plot(x,z)
```

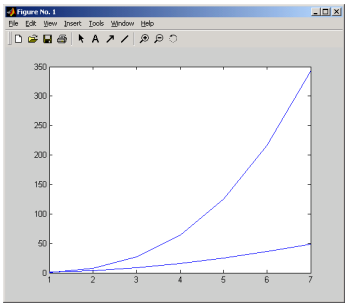


***Figura3**. Ejemplo de construcción de gráficas superpuestas.*

Matlab permite representar varias subfiguras en una única figura. Para ello, se emplea el comando **subplot(m,n,p)** donde el primer parámetro corresponde al número de filas, el segundo parámetro al número de columnas y el tercer parámetro a la posición relativa en la matriz de subfiguras.

Por ejemplo, si se desea representar tres señales en una misma fila se ejecutarían las siguientes líneas de comandos:

```
>> subplot(1,3,1)
>> plot(señal1)
>> subplot(1,3,2)
>> plot(señal2)
>> subplot(1,3,3)
>> plot(señal3)
```

3.3 Ejemplo de representación de una gráfica.

Dibujar la gráfica de la función sin(x) en el intervalo $0 \le x \le 10$ con un paso de 0.05.

```
>>x=0:0.05:10;
>>y=sin(x);
>>plot(x,y);
>>grid;
>>title('Gráfica del seno');
>>xlabel('Segundos');
>>ylabel('Seno(x)');
>>text(3,0.45,'sen(x)');
```

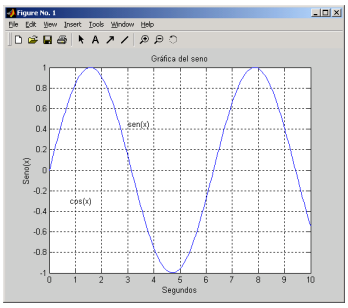


***Figura 4**. Ejemplo de representación gráfica y etiquetado.*

4. Scripts y funciones.

Un **script** es un conjunto de instrucciones (de cualquier lenguaje) guardadas en un fichero (usualmente de texto) que son ejecutadas normalmente mediante un intérprete. Son útiles para automatizar pequeñas tareas. También puede hacer las veces de un "programa principal" para ejecutar una aplicación.

Así, para llevar a cabo una tarea, en vez de escribir las instrucciones una por una en la línea de comandos de MATLAB, se pueden escribir una detrás de otra en un fichero. Para ello se puede utilizar el Editor integrado: icono "hoja en blanco" del menú de herramientas, opción "New M-file" del Menú "File" o bien usando la orden

```
>> edit
```

Los scripts de MATLAB deben guardarse en un fichero con sufijo .m para ser reconocidos. Para ejecutar un script que esté en el directorio de trabajo, basta escribir su nombre (sin el sufijo) en la línea de comandos.

Una **función** (habitualmente denominadas M-funciones en MATLAB), es un programa con una "interfase" de comunicación con el exterior mediante argumentos de entrada y de salida. Las funciones MATLAB responden al siguiente formato de escritura:

function [argumentos de salida] = nombre(argumentos de entrada)
% comentarios
....
instrucciones (normalmente terminadas por ; para evitar eco en pantalla)
....

5. Transformadas y transformadas inversas.

La transformada de Laplace, la transformada de Fourier y la transformada z son tres técnicas de transformación que proporcionan una conversión de variables. Todas ellas convierten modelos de ecuaciones diferenciales lineales a modelos algebraicos. La transforma de Laplace se usa para obtener una función de transferencia, que modeliza el comportamiento de un sistema continuo, mientras que la transformada z modeliza el comportamiento de sistemas discretos.

Nombre de función	Comentario
`fourier`	Transformada de Fourier
`ifourier`	Transformada inversa de Fourier
`laplace`	Transformada de Laplace
`ilaplace`	Transformade inversa de Laplace
`ztrans`	Transformada z
`iztrans`	Transformada z inversa

***Tabla 3**. Funciones de transformación.*

A continuación se presentan varios ejemplos que ayudan a familiarizarse con la sintaxis de las órdenes mostradas en la tabla 3.

Si se desea calcular por ejemplo la transformada de Laplace de una función $f(t) = sen(t) - 2e^{-2t}$; en primer lugar se necesita declarar una variable de tipo simbólica con el uso de la instrucción `syms` y después se ejecuta el comando de la transformada que se desea calcular pasando como parámetro la función de la cual se quiere calcular su transformada. Por ejemplo, para este caso:

```
>> syms t
>> TL= laplace(sin(t)-2*exp(-2*t))
```

El resultado que se obtiene es:

```
TL =
1/(s^2+1)-2/(s+2)
```

Si ahora se calcula la transformada inversa de Laplace se obtiene la función f(t) de partida:

```
>> syms s
>> f=ilaplace(1/(s^2+1)-2/(s+2))

f =
 sin(t)-2*exp(-2*t)
```

En el caso de un sistema discreto, se puede obtener la transformada z de un modo muy similar:

```
>> syms n
>> TZ= ztrans((-9*(0.9)^n+10)
```

El resultado que se obtiene es :

```
TZ =
(10*z)/(z - 1) - (9*z)/(z - 9/10)
```

Si ahora se calcula la transformada z inversa se obtiene la función f(n) de partida:

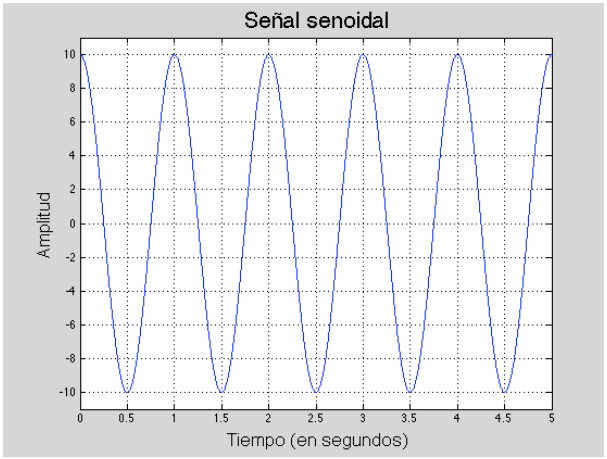
```
>> syms z
>> f=iztrans((10*z)/(z - 1) - (9*z)/(z - 9/10))

f =
10 - 9*(9/10)^n
```

Ejercicios propuestos

Ejercicio 1

Apartado a

Realizar un script de Matlab llamado **senoidal.m** que represente una señal senoidal en función de los siguientes parámetros de entrada: tiempo (T), frecuencia (F), amplitud (A) y fase (Ø). Los parámetros de entrada deberán especificarse al principio del script. Este es el resultado que debe mostrar la ejecución del script:

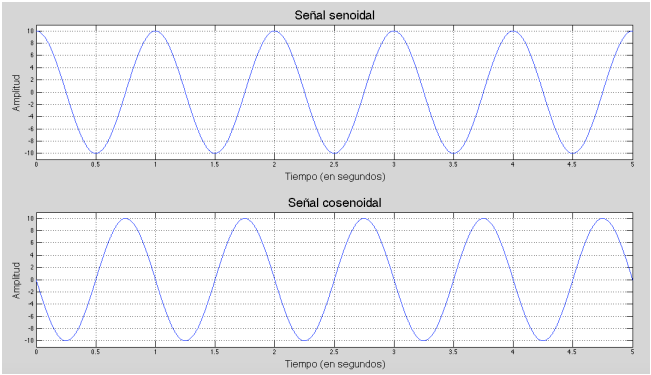


A continuación crear una función llamada **fsenoidal.m** que responda a la siguiente estructura de llamada:

```
>> fsenoidal(frecuencia, amplitud, fase, tiempo)
```

Apartado b

Realizar un script de Matlab **senyales.m** que represente una señal senoidal y una señal cosenoidal en una misma figura. Los parámetros de ambas señales se definirán al principio del script: tiempo (T), frecuencia (F), amplitud (A) y fase (Ø). El aspecto de la ejecución debe ser el siguiente:



Apartado c

Crear una función llamada **ftipo.m** que represente una señal de tipo senoidal, cosenoidal o triangular en función de los siguientes parámetros de entrada: tipo de señal, tiempo (T), frecuencia (F), amplitud (A) y fase (Ø).

```
>> ftipo(tipo, frecuencia1, amplitud1, fase1, tiempo1)
```

Ayuda: se sugiere utilizar un condicional **switch** para discriminar el tipo de señal y la función **sawtooth** para generar la señal triangular

Ejercicio 2

La transformada de Laplace es una herramienta matemática más usada en control. Ésta permite obtener una solución simbólica que simplifica las ecuaciones físicas que modelan el comportamiento del sistema continuo que se quiere someter a estudio. Aplicando transformaciones como la de Laplace se consiguen funciones donde las relaciones causa-efecto en un sistema son más fáciles de entender. Así, en definitiva, con la transformada de Laplace lo que se consigue es simplificar las funciones que modelan el sistema convirtiendo ecuaciones diferenciales de variable real en modelos algebraicos de variable compleja. En el caso de control de sistemas discretos, se utiliza la transformada Z.

Apartado a

Determinar las transformadas de Laplace de estas funciones de *t*.

$$x(t) = \left[2 + 3t + 4t^2\right] u(t)$$
$$x(t) = \left[2 - 2e^{-4t} + 3te^{-4t}\right] u(t)$$
$$x(t) = \left[4sen(2t) + 5cos(2t)\right] u(t)$$
$$x(t) = (2t)u(t-1)$$

Ayuda: para la función escalón utilizar el comando **heaviside(t-a)** donde a es el retraso

Apartado b

Determinar las transformadas Z de estas funciones de *n*.

$$x(n) = (1/4)^n u(n)$$
$$x(n) = (1/2)^n u(n) + (-1/3)^n u(n)$$
$$x(n) = \left[2 + 5e^{-2n}\right] u(n)$$
$$x(n) = (1/4)^n u(n-1)$$

Ayuda: para la función escalón utilizar el comando **heaviside(n-a)** donde a es el retraso

PRACTICA 1 – Análisis de sistemas en el dominio del tiempo
Sesión 1 – Función de transferencia, diagrama de bloques y modelado de sistemas

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

1. Modelización de sistemas usando MATLAB.

MATLAB presenta un conjunto de rutinas útiles para transformar un determinado modelo matemático de un sistema lineal en otro modelo. A continuación, se presenta la metodología empleada para aplicar estas transformaciones de modo que sirvan para resolver problemas de ingeniería de control.

1.1 Funciones de transferencia. Sistemas continuos.

La función de transferencia se describe como un cociente de dos polinomios. Un modo de introducir una función de transferencia en MATLAB consiste en definir dos vectores fila, cada uno de los cuales contendrá los coeficientes del numerador y los del denominador, respectivamente.

Por ejemplo, para definir la función de transferencia:

$$G(s) = \frac{Y(S)}{U(s)} = \frac{5s + 20}{s^2 + 4s + 20}$$

se realiza en MATLAB la siguiente secuencia de comandos:

```
>> n=[0 5 20]
>> d=[1 4 20]
```

En control, existen diversas técnicas de análisis y diseño que necesitan obtener las raíces de los polinomios numerador y denominador de la función de transferencia para conocer la localización de sus polos y ceros, así como una constante denominada factor de ganancia. Los polos se pueden definir matemáticamente como las raíces del polinomio del denominador y los ceros como las raíces del numerador.

MATLAB proporciona una orden de conversión para transformar la función de transferencia a una notación que permita trabajar con los polos y ceros y ganancia por separado. Por ejemplo, si se quiere pasar la función de transferencia anterior a polos-ceros bastará con ejecutar el comando:

```
>> [z p k]=tf2zp(n,d)
z =
    -4
p =
  -2.0000 + 4.0000i
  -2.0000 - 4.0000i
k =
     5
```

donde z es un vector que contiene los valores de los ceros, p los valores de los polos y k el factor de ganancia.

Además, también se puede realizar la operación contraria. Así, dados los valores de polos-ceros-ganancia se puede obtener la función de transferencia equivalente ejecutando el comando *zp2tf*. Por ejemplo, se conocen los ceros $(s+4)$ y los polos $(s+2-4j)(s+2+4j)$ y se quiere obtener la función de transferencia equivalente.

```
>> k=5
>> z=-4
>> p=[-2+4*j -2-4*j]'
>> [n,d]=zp2tf(z,p,k)

n =
     0     5    20
d =
     1     4    20
```

Otra manera de introducir la función de transferencia en dominio de Laplace es haciendo uso de la orden tf.

```
>> G=tf(n,d)
```

Otro ejemplo:

```
>> G=tf([1 1],[1 2 1])
```

Transfer function:

```
     s + 1
-------------
s^2 + 2 s + 1
```

De este modo la variable G representará una vez ejecutada la orden *tf* el sistema continuo identificado por su función de transferencia.

1.2 Descomposición de la función de transferencia.

La descomposición en fracciones simples es una metodología matemática muy útil para obtener los polos y ceros cuando la función de transferencia está constituida por polinomios de un orden elevado.

Por ejemplo, dada la función de transferencia $G(s)=\frac{Y(s)}{U(s)}=\frac{2s^3+5s^2+3s+6}{s^3+6s^2+11s+6}$ se quiere obtener su descomposición en fracciones simples.

El primer paso es introducir la función de transferencia:

```
>> n=[2 5 3 6]
>> d=[1 6 11 6]
>> printsys(n,d)

num/den =

   2 s^3 + 5 s^2 + 3 s + 6
   -----------------------
    s^3 + 6 s^2 + 11 s + 6
```

Una vez se ha introducido ésta en forma de dos vectores fila, se utiliza la orden *residue* de la siguiente manera:

```
>> [r p k]=residue(n,d)

r =
   -6.0000
   -4.0000
    3.0000
p =
   -3.0000
   -2.0000
   -1.0000
k =
    2
```

La salida p es un vector columna que contiene los polos, la salida r es otro vector columna que contiene los residuos de los polos correspondientes y la salida k es un vector fila que contiene los términos constantes.

Así, el resultado de descomponer la función de transferencia anterior en fracciones simples es:

$$G(s)=\frac{Y(s)}{U(s)}=\frac{-6}{(s+3)}+\frac{-4}{(s+2)}+\frac{3}{(s+1)}+2$$

También, es posible pasar de un conjunto de fracciones parciales simples al cociente polinómico que determina una función de transferencia, con la función *residue*, como sigue:

```
>> [n,d]=residue(r,p,k);

n =
    2.0000    5.0000    3.0000    6.0000
d =
    1.0000    6.0000   11.0000    6.0000
```

1.3 Interconexión de bloques.

La función de transferencia de un sistema es un modelo algebraico que relaciona variables de salida con variables de entrada. Un diagrama de bloques se puede entender como una representación gráfica de esas relaciones algebraicas. MATLAB permite el cálculo de la función de transferencia en bucle cerrado de un sistema representado por un diagrama de bloques. Para reducir un diagrama de bloques a su función de transferencia equivalente se pueden usar las funciones *feedback*, *series* o *parallel*.

A continuación, se presenta la secuencia de comandos que permiten calcular la función de transferencia en bucle cerrado del sistema representado en la figura 1.

```
>> G=tf([0 0 100],[1 2 0])    %Funcion G(s)
>> H=tf([0 1],[1 20])    %Funcion H(s)
%Calcula función de transferencia de la realimentación
>> M=feedback(G,H,-1)  % Funcion M(s)
```

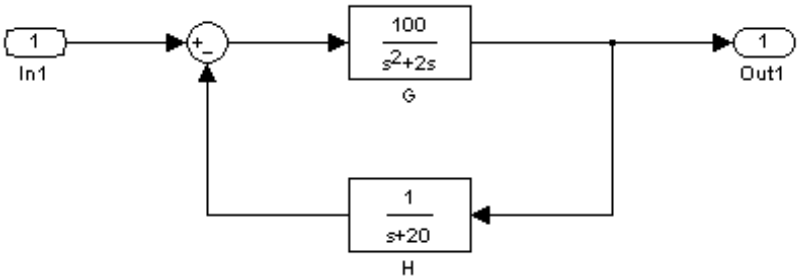


***Figura 1**. Diagrama de bloques equivalente.*

Si se aumenta la complejidad del sistema hasta el representado por la figura 2. La secuencia de ordenes se podría poner como:

```
>> K=tf(0.4,1)  % Ganancia K
>> G=tf([0 0 100],[1 2 0]) %Funcion G(s)
>> H=tf([0 1],[1 20])  % Funcion H(s)
%Calcula función de transferencia de la realimentación G y H
>> R1=feedback(G,H,-1)
%Calcula función de transferencia en serie con el factor K
>> S1=series(K,R1)
%Calcula función de transferencia de la realimentación general
>> M=feedback(S1,tf(1,1),-1) % Funcion M(s)
```

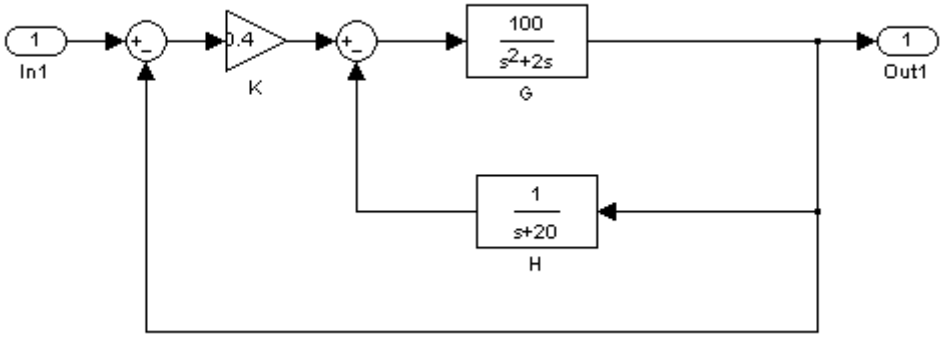


***Figura 2**. Diagrama de bloques.*

1.4 Visualización de la respuesta de un sistema

Para ilustrar el comportamiento de un sistema continuo se puede analizar su respuesta ante, por ejemplo, una entrada escalón (en las siguientes sesiones, se estudiará en más detalle el comportamiento temporal de sistemas). Por ejemplo, si se tiene un sistema continuo con una función de transferencia G, se puede obtener su respuesta ante entrada escalón con la función step del siguiente modo:

```
>> n=[0 5 20];
>> d=[1 4 20];

>> G=tf(n,d);

>> t=10 % tiempo de representación
>> [Y,T] = step(G,t); % respuesta ante escalón

>> plot(T,Y)
>> grid
>> title('Respuesta ante entrada escalón')
>> xlabel('Tiempo(en segundos)')
```

El resultado mostrado será el siguiente:

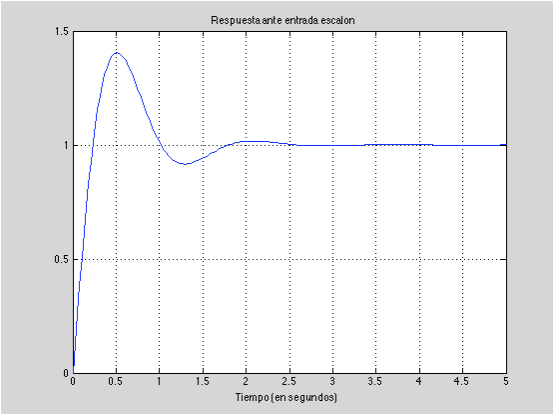


***Figura 3**. Respuesta ante entrada escalón de un sistema continuo*

2. Modelización de sistemas usando SIMULINK.

MATLAB presenta una interfaz gráfica basada en ventanas llamada SIMULINK. La ventaja que ofrece SIMULINK es la posibilidad de modelar sistemas de modo más intuitivo, como es el modelado gráfico, sin necesidad de utilizar la ventana de línea de comandos utilizada hasta el momento. La potencia y versatilidad de este entorno gráfico se comprueba sobre todo a la hora de describir gráficamente un sistema dibujando su diagrama de bloques directamente, es decir, sin utilizar funciones como *feedback*, *series* o *parallel* que hasta ahora modelaban mediante comandos ese mismo diagrama de bloques.

2.1 Iniciar SIMULINK.

Para iniciar SIMULINK se puede ejecutar la orden simulink en la ventana de comandos de MATLAB o bien pinchar en el botón  . Se abrirá una biblioteca con todos los elementos (bloques de funciones de transferencia, tipos de entradas, sumadores, bloques de factores de ganancia, etc) necesarios para dibujar diagramas de bloques (figura 4). Todos estos bloques se organizan en diferentes grupos de acuerdo a cuál es su funcionalidad y cometido. Algunos de los grupos más empleados son:

- **Continuous:** dispone de bloques relacionados con el análisis de sistemas continuos, como funciones de transferencia en el dominio s.
- **Discrete:** dispone de bloques relacionados con el análisis de sistemas discretos, como funciones de transferencia en el dominio z.
- **Sources:** dispone de los bloques de generación de señales, por ejemplo, para generar una función escalón o rampa.
- **Sinks:** dispone de bloques que sólo reciben señales, sin producir una salida, por ejemplo, para poder visualizar la salida de un sistema.
- **Math operations:** dispone de bloques para realizar operaciones matemáticas en el flujo de señales, por ejemplo, un sumador.

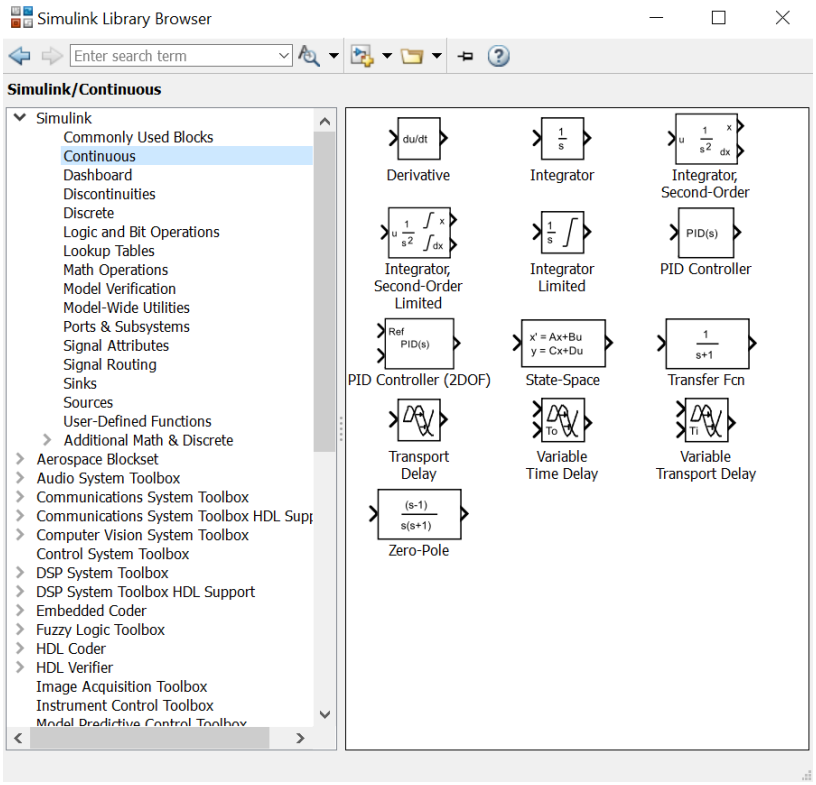


Figura 4. Aspecto de la biblioteca de Simulink

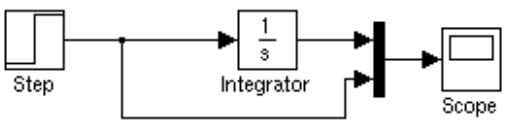
2.2 Construir un diagrama de bloques.

Para dibujar un diagrama de bloques primero hay que crear un nuevo modelo: *File* -> *New* -> *Blank_Model*. A partir de ahí sólo se requiere la utilización del ratón con el que se pueden arrastrar y pegar los bloques de las distintas bibliotecas. Cuando el elemento se encuentre sobre la ventana de trabajo soltando el botón del ratón éste queda fijo. Mediante este procedimiento de pinchar, arrastrar y soltar se pueden seleccionar los elementos deseados para la construcción de un determinado diagrama de bloques.

A continuación, se muestra un ejemplo de un diagrama de bloques sencillo con un integrador y una señal de entrada escalón unitario.

Elementos:
Entrada escalón (*Sources* -> *Step*)
Visor (*Sinks* -> *Scope*)
Integrador (*Continuous* –> *Integrator*)
Multiplexor (*Signal Routing* -> *Mux)*

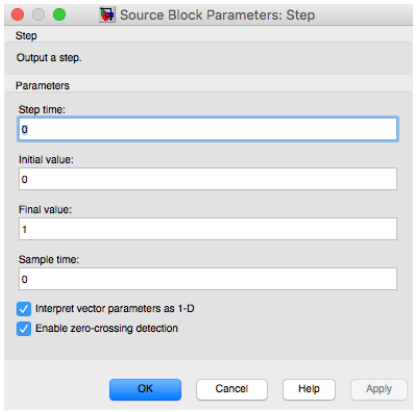
Esquema:



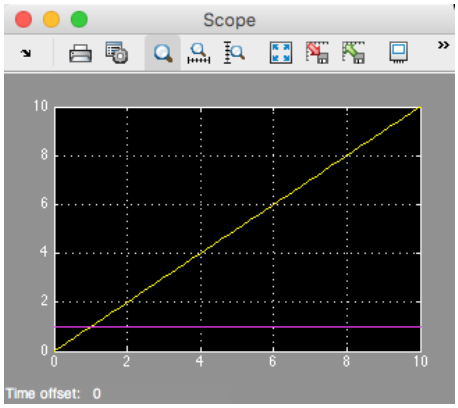
Después de haber escogido los elementos deseados, el siguiente paso es introducir los valores de esos elementos, para que empiecen a ser parte activa del diagrama de bloques que modela el sistema a implementar. Así, de acuerdo al tipo de elemento que representa cada bloque se

podrá introducir un determinado valor. En este caso, se pueden definir las características del escalón unitario pinchando dos veces en el bloque (figura 5, izda).

Por último, mediante la opción *Simulation -> Start*, se puede simular el comportamiento del sistema. La salida muestra tanto la entrada (escalón) como la salida del integrador (figura 5, dcha).



Parámetros del bloque Step



Respuesta resultante de la simulación anterior

Figura 5. Comportamiento del sistema integrador

2.3 Funciones de transferencia (lazo abierto y lazo cerrado)

Para utilizar funciones de transferencia en SIMULINK se debe incluir un bloque Continuous -> Transfer Function. Como antes, los parámetros de dicho bloque se modifican pulsando dos veces sobre el bloque. En el cuadro de diálogo que aparece, se introducen los polinomios numerador y denominador, siguiendo la sintaxis seguida en la Sesión 1 de esta práctica. En el siguiente esquema se puede ver un ejemplo:

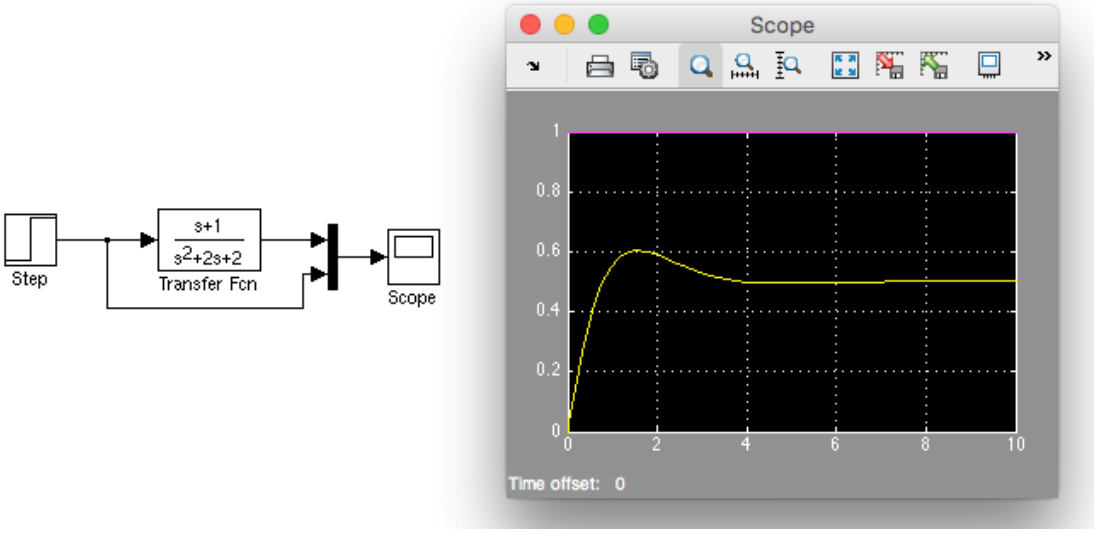


Figura 6. Sistema en lazo abierto

El ejemplo anterior, muestra un sistema en lazo abierto, es decir, no existe una realimentación de la salida a la entrada del sistema. Para introducir una realimentación se debe usar el bloque *Math Operations -> Sum*. Este bloque puede variar su signo si se trata de una realimentación positiva y negativa. Adicionalmente, se pueden introducir ganancias mediante el bloque *Math Operations -> Gain*. En el siguiente ejemplo se muestra un sistema con todos estos elementos:

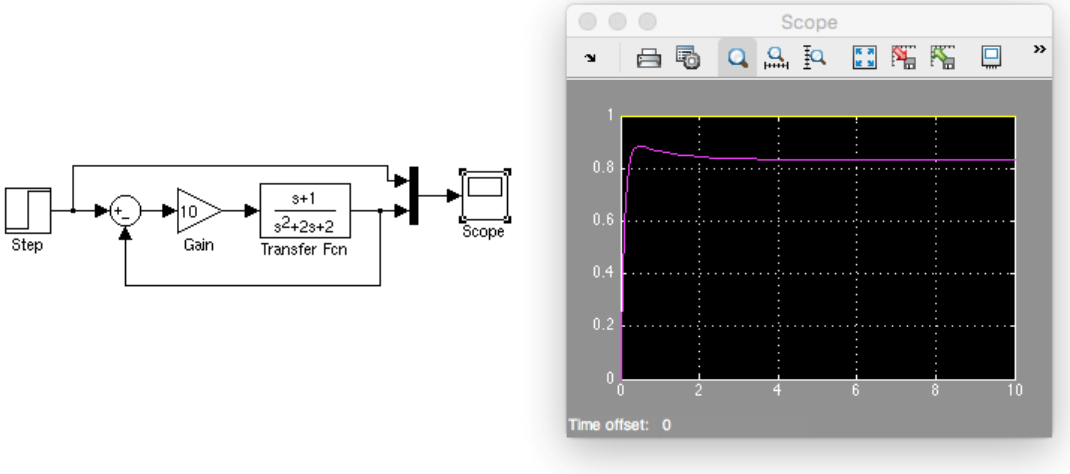


Figura 7. Sistema en lazo cerrado

2.4 Fuentes de señal (sources) y visualización (sinks).

Hemos visto un ejemplo de cómo introducir una entrada escalón unitario al sistema. SIMULINK dispone de distintos elementos que pueden actuar como señal de entrada del sistema. Estos elementos se encuentran en la sección Source. Los elementos de este apartado son, esencialmente, generadores de señales predeterminadas: señal de valor constante (*Constant*), señal escalón (*Step*), señal rampa (*Ramp*), señales senoidales (*Sine Wave*) y muchas otras. También permite guardar el tiempo de simulación mediante el bloque denominado Clock, que proporciona un vector con los instantes de simulación. Otra opción es introducir señales provenientes directamente del workspace de MATLAB. Esta es una de las grandes ventajas de interacción entre MATLAB y SIMULINK.
En el otro extremo, tenemos elementos que sólo reciben señales, sin producir ninguna salida. Estos elementos se encuentran en el apartado *Sinks*. Entre estos elementos, encontramos los visores, como el *Scope* ya empleado. Otro elemento interesante de este grupo es el bloque *To Workspace*. Se trata de un bloque que almacena, en una variable cuyo nombre debemos especificar, un vector conteniendo los valores que va tomando la señal que se recibe como entrada del bloque, para cada instante de simulación. Permite, por tanto, guardar y procesar posteriormente los resultados de la simulación.

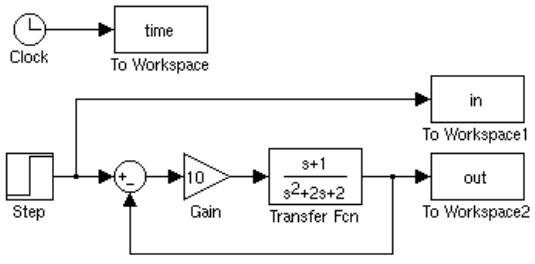


Figura 8. Almacenamiento de señales en el Workspace

2.5 Simulación.

En los apartados anteriores ya se ha visto como simular el sistema mediante la opción *Simulation -> Run*. En SIMULINK es posible modificar los parámetros de simulación en función

de nuestras necesidades. Para ello, se accede a los parámetros de simulación en el menú contextual *Simulation -> Model Configuration Parameters*. Entre ellos cabe destacar, el tipo de algoritmo de simulación, por ejemplo, **Limsim** para este caso, el tiempo de comienzo **Start Time**, el tiempo de parada Stop Time, el tamaño mínimo del escalón **Min Step Size**, el tamaño máximo del escalón **Max Step Size** y la tolerancia **Tolerance**.

Para guardar un modelo se escoge la opción **Save** o **Save as** en el menú *File*. SIMULINK guarda el modelo generando un archivo con extensión *.mdl* o *.slx* que contiene las ordenes de MATLAB necesarias para reconstruir el modelo. Para abrir un modelo se escoge la opción **Open** del menú *File*. Una vez guardado el diagrama de bloques, éste puede volver a ser llamado en sesiones futuras simplemente introduciendo el nombre con el que se guardó sobre la línea de comandos de MATLAB.

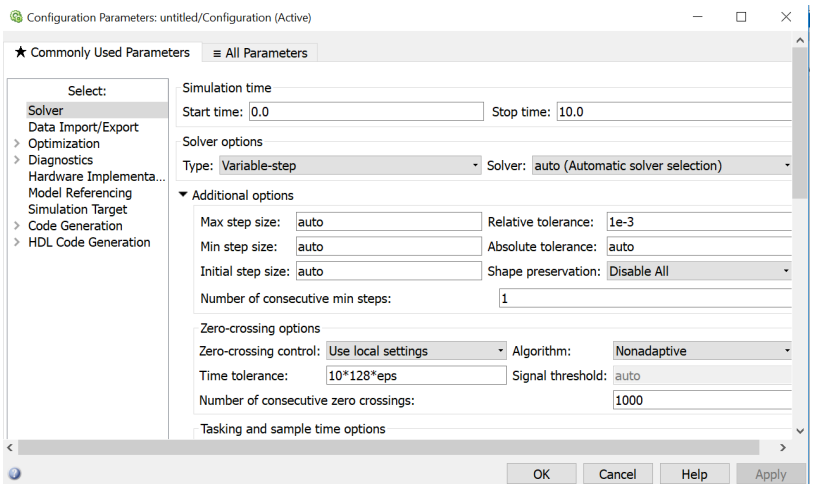


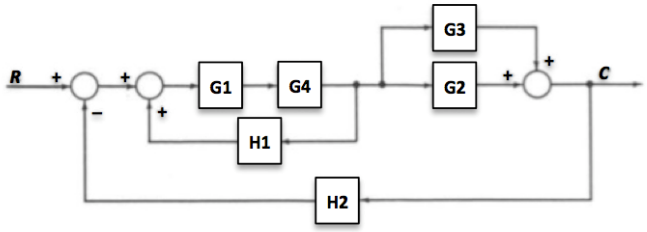
Figura 9. Parámetros de la simulación

Ejercicios propuestos

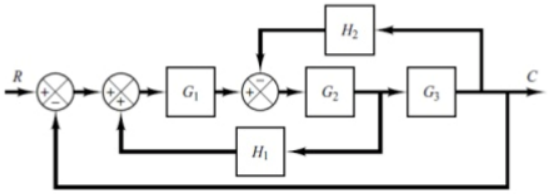
Ejercicio 1

Calcula, utilizando los comandos de Matlab *feedback*, *series* y *parallel*, la función de transferencia equivalente de los siguientes diagramas de bloques:
Nota: *en alguno de los ejercicios se deberá realizar algún cambio previo para poder resolverlo*

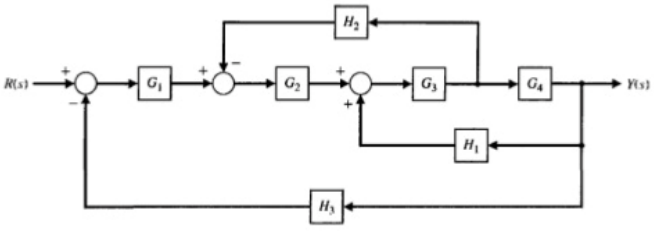
G1=3
G2=1/s
G3=1/(s+3)
G4=2
H1=1
H2=1/2



G1=2
G2=1/(s+1)
G3=1/s
H1=1/2
H2=1/2



G1=4
G2=1/(s+2)
G3=2
G4=1/s
H1=1/2
H2=1
H3=1/4



Ejercicio 2

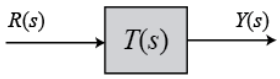
A partir de los diagramas propuestos en el ejercicio anterior. Se pide:

Apartado a
Simular su comportamiento en Simulink y obtener la salida se los sistemas ante una entrada escalón unitario.

Apartado b
Simular el comportamiento del sistema reducido en Simulink (función de transferencia obtenida previamente) y comprobar si la reducción del sistema es correcta mediante la visualización de su salida ante entrada escalón.

3. Modelado de sistemas físicos. Implementación en Simulink.

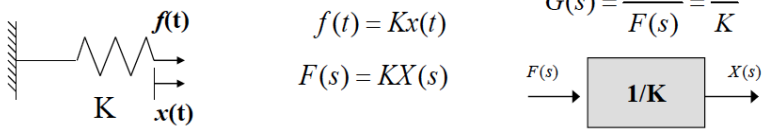
Como ya hemos visto en los apartados anteriores, una función de transferencia es un modelo matemático que a través de un cociente relaciona la respuesta de un sistema Y(s) con una señal de entrada o excitación R(s).



En la teoría de control, a menudo se usan las funciones de transferencia para caracterizar las relaciones de entrada y salida de componentes o de sistemas que se describen mediante ecuaciones diferenciales lineales e invariantes en el tiempo.

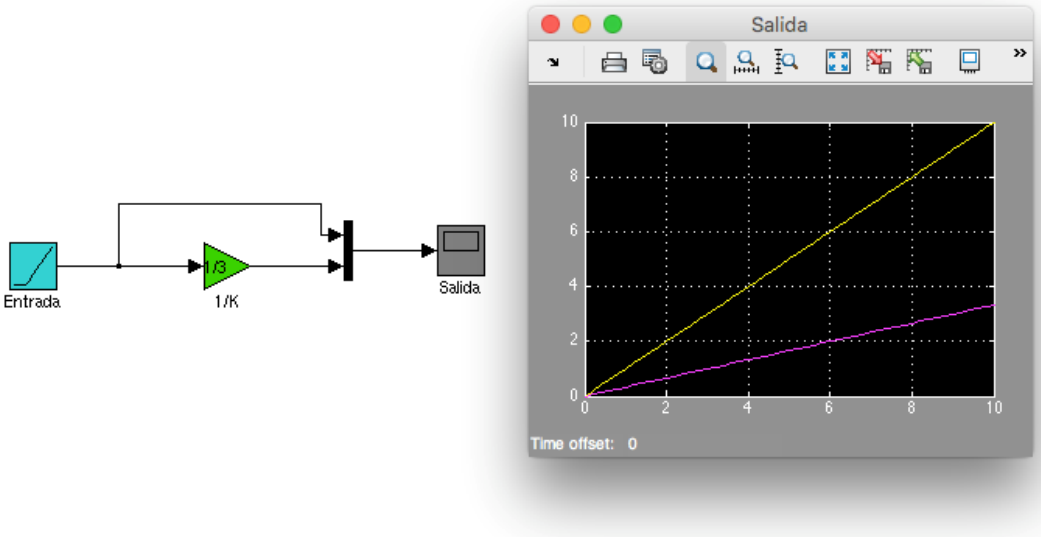
Ejemplo de sistema mecánico

Por ejemplo, si se quiere modelar el comportamiento de un muelle:



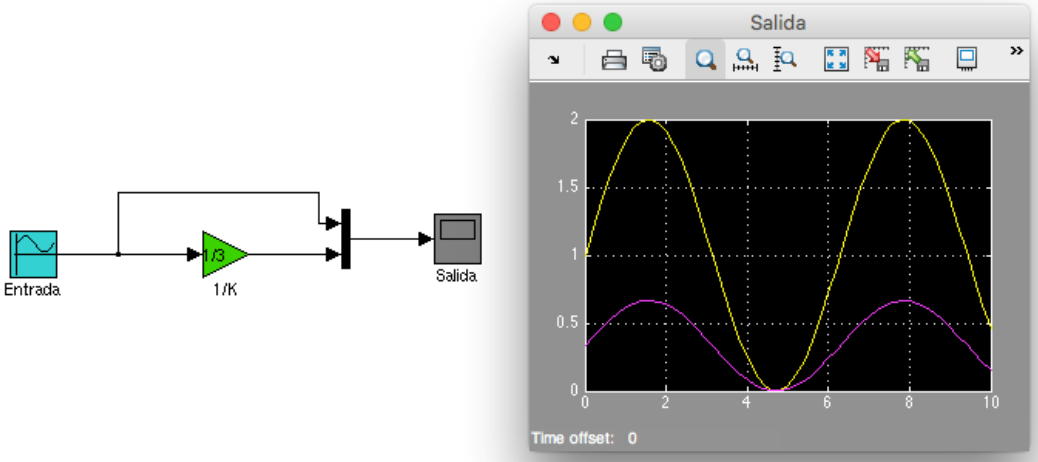
Tomando como entrada del sistema la fuerza ejercida sobre el muelle (F) y como salida el desplazamiento del mismo (X), se observa que la fuerza es proporcional al desplazamiento en función de su constante elástica (K). Si se transforma al dominio de Laplace se obtiene que el sistema puede definirse por una ganancia de valor 1/K.

Si se quisiera modelar este comportamiento mediante Simulink, de forma muy simple, se debería dibujar el siguiente diagrama:



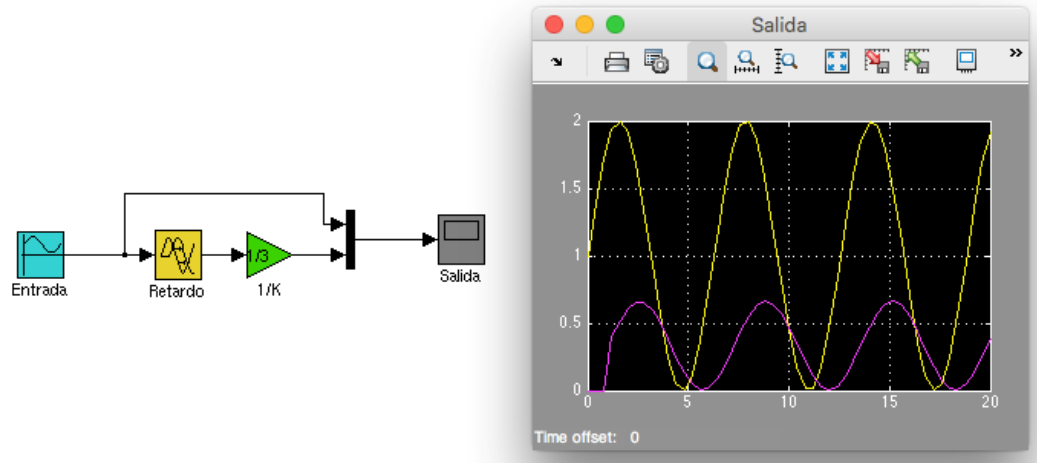
donde la constante elástica K es 3 N/m. Por tanto, ante una entrada rampa unitaria (incremento de fuerza constante en newtons, N) (señal amarilla) se obtendría una salida (desplazamiento en metros, m), tres veces más pequeña que la fuerza ejercida (señal rosa).

Ahora, vamos a suponer que, en lugar de un incremento constante, se aplica una fuerza variable al muelle, como una senoidal con una amplitud de pico de 2 N y período de 1 segundo:



Como se puede observar una aplicación de fuerza oscilatoria como entrada se traduce en un desplazamiento oscilatorio del muelle. En este caso, la fuerza aplicada nunca es negativa puesto que el muelle, en reposo, sólo permite su elongación en una dirección.

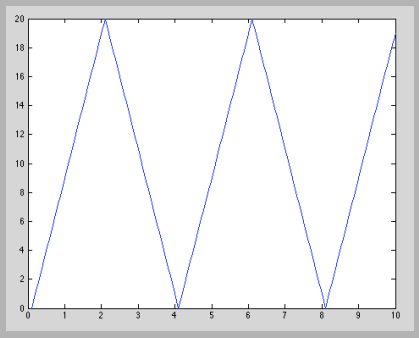
Vamos a suponer ahora, que el efecto de aplicar fuerza en el muelle tiene cierto retardo sobre el desplazamiento, debido a las propiedades mecánicas del mismo. Una forma de modelar este retardo se muestra a continuación:



En este caso, el muelle se alargará con un retardo de 1 segundo respecto a la aplicación de la fuerza.

Generación manual de entradas en Simulink

En Simulink, es posible generar entradas al sistema de distintos tipos. Además de las ya conocidas: escalón o rampa; es posible generar entradas periódicas e incluso, generar entradas predefinidas por el usuario. Para generar entradas definidas manualmente se puede utilizar el bloque From Workspace de la biblioteca Sources, que permite leer cualquier vector, matriz o estructura de datos disponible en el espacio de trabajo de Matlab. Por ejemplo para generar la siguiente señal triangular:



Se definiría la señal de entrada (input) como:

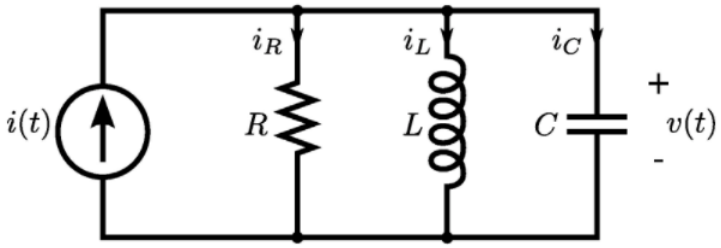
```
input=[0.1:0.1:10;[0:20 19:-1:0 1:20 19:-1:0 1:19]]'
```

De este modo, se pueden definir señales de todo tipo e incluso leer señales obtenidas de otros procesos ejecutados previamente.

Además de calcular la función de transferencia global entre la entrada y la salida final del sistema, es posible generar un diagrama de bloques desglosado a partir de las ecuaciones físicas del mismo.

Ejemplo de sistema eléctrico

Por ejemplo, en el caso de un circuito RLC en paralelo mostrado a continuación:



Resolviendo por nodos:

$$i_s = i_R + i_L + i_C$$

donde:

$$i_R = \frac{V}{R} \qquad i_L = \frac{1}{L}\int V(t)dt \qquad i_C = C\frac{dV}{dt}$$

sustituyendo se obtiene:

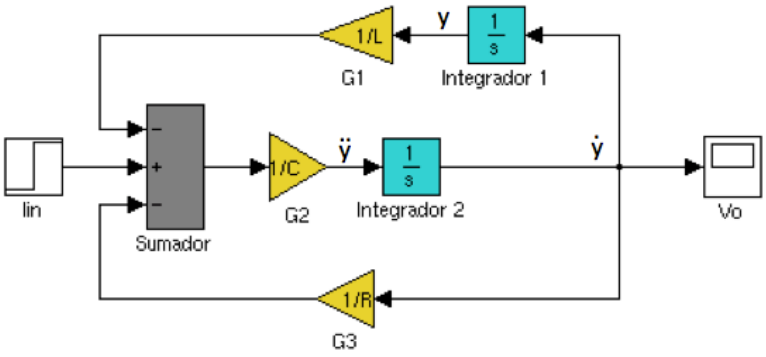
$$i_s = \frac{V}{R} + \frac{1}{L}\int V(t)dt + C\frac{dV}{dt}$$

una vez obtenida la ecuación diferencial que define el sistema se puede construir el diagrama de bloques que relaciona las variables de entrada x (intensidad) con las variables de salida y (voltaje). Para ello, transformamos a otra notación simbólica las variables derivadas e integradas de la ecuación. Este paso, permite resolver de forma más intuitiva el diagrama:

$$x = \frac{1}{R}\dot{y} + \frac{1}{L}y + C\ddot{y}$$

donde cada punto representa una derivación sobre la variable.

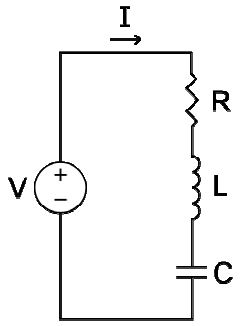
Con esta notación, es posible construir el siguiente esquema de Simulink:



Ejercicios propuestos

Ejercicio 3

Dado el circuito RLC en serie que se muestra a continuación:



Se pide:

1. Modelar el sistema y obtener la función de transferencia G(s) que relaciona voltaje de entrada V(s) y la intensidad de la malla I(s).
2. Evaluar el comportamiento del sistema mediante Simulink ante una entrada escalón para un valor de R=1, C=1 y L=1.
3. Obtener el diagrama de bloques desglosado que permita visualizar los voltajes específicos de cada componente: V_R, V_L y V_C.
4. Modelar los siguientes casos particulares y estudiar la respuesta del sistema (mantener los valores R, L y C iniciales según sea el caso):
    - Circuito R, sólo existe la resistencia
    - Circuito RL (resistencia más bobina)
    - Circuito RC (resistencia más condensador)

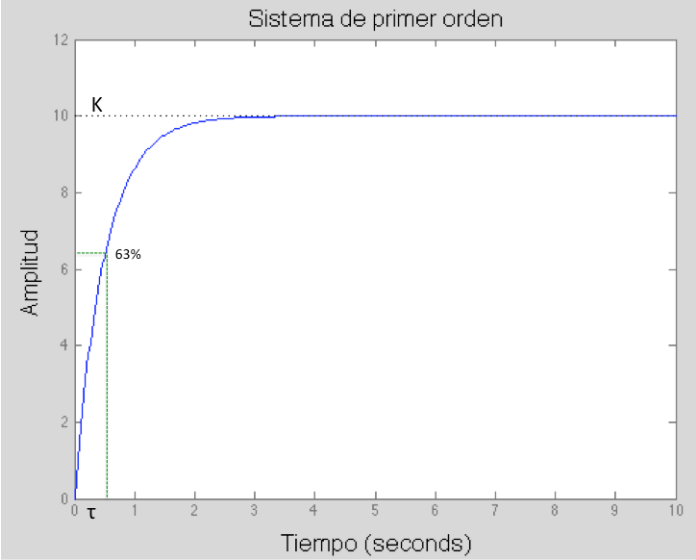
PRACTICA 1 – Análisis de sistemas en el dominio del tiempo
Sesión 2 - Sistemas de primer orden

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

En esta práctica, se analizará el comportamiento en el dominio del tiempo de los sistemas de primer orden. Se denominan sistemas de primer orden a aquellos en los que en la ecuación general se reduce al formato siguiente:

$$\tau \frac{dy}{dt} + y = k\,u$$

La respuesta típica de estos sistemas no presenta sobreoscilación, esto quiere decir que nunca llegan al valor exacto de la consigna y por lo tanto, son sistemas relativamente lentos. Un ejemplo de ello es el calentamiento de un horno.



En el dominio de Laplace, los sistemas de primer orden están definidos por la siguiente función de transferencia:

$$G(s) = \frac{K}{1 + \tau \cdot s}$$

donde K es la ganancia del sistema y τ es la constante de tiempo.

El valor de la ganancia se calcula mediante la siguiente fórmula:

$$K = \frac{Señal\ salida}{Señal\ entrada} = \frac{\Delta y}{\Delta u}$$

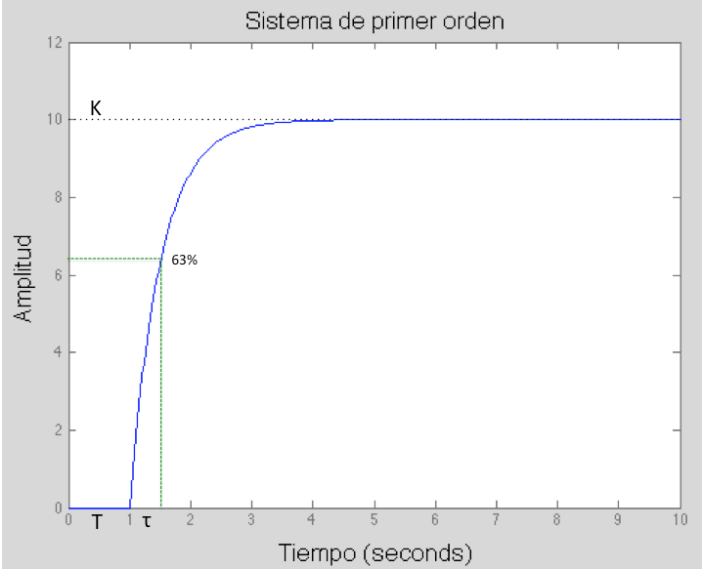
El valor de la constante de tiempo se obtiene sobre la gráfica, para ello se observa el tiempo correspondiente a un valor del 63% Δy.

En ocasiones, también se trabaja con un factor denominado tiempo de establecimiento, que suele estar comprendido entre un 95–98%. Este factor determina el tiempo en el cual la respuesta se estabiliza entre los límites indicados a ese porcentaje.

Un caso particular de los sistemas de primer orden, es un sistema de primer orden con retardo. En este caso, la función de transferencia sería del siguiente modo:

$$G(s) = \frac{K}{1+\tau \cdot s} \cdot e^{-T \cdot s}$$

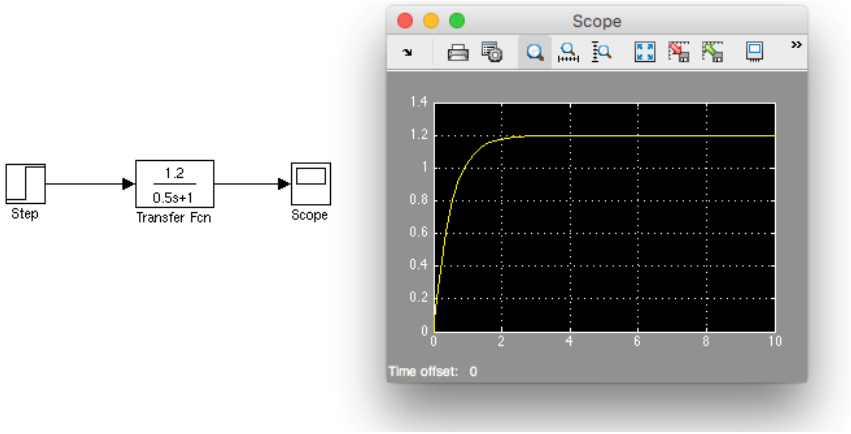
donde T es el retardo en la respuesta del sistema. El aspecto de esta respuesta puede verse a continuación:



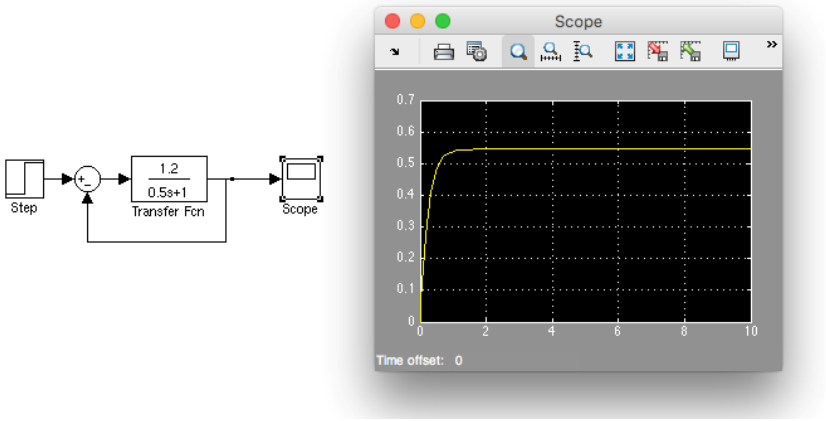
Sistemas de primer orden realimentados

En el caso de tener un sistema de primer orden con una ganancia K y una constante de tiempo τ determinadas, en lazo abierto, nuestra salida va a tener un valor únicamente dependiente de los parámetros intrínsecos del sistema, es decir, no vamos a poder saber cuál va a ser la amplitud de su respuesta y si esta respuesta se va a ajustar a nuestra consigna. En el ejemplo de un horno, que es un típico sistema de primer orden, el valor de la temperatura final será desconocido y dependerá del comportamiento del sistema ante una consigna determinada, es decir, ante la selección de una determinada temperatura deseada.

Por ejemplo, en el siguiente sistema de primer orden en lazo abierto, la salida alcanza un valor de 1.2 ante entrada escalón unitario, debido a que la ganancia K del sistema es 1.2. Esta salida sólo depende de la ganancia del sistema y por tanto, no se ajusta a la consigna introducida.



Para poder controlar que la salida se ajuste a nuestros requerimientos se debe cerrar el bucle. Siguiendo el caso anterior se obtiene:



La entrada de control del sistema en lazo cerrado depende de la salida obtenida en cada momento, por lo que el sistema se reajusta para poder alcanzar la consigna. No obstante, ante la ausencia de un controlador que ajuste los parámetros del sistema, se produce un error entre la amplitud de salida y de entrada que se denomina error de posición en régimen permanente (e_p) y se puede calcular del siguiente modo:

$$e_{p_\infty} = \frac{1}{1+K_p}; \quad K_p = \lim_{s\to 0} G(s)$$

En prácticas posteriores se verá en más detalle todos los aspectos relacionados con los errores del sistema en régimen permanente.

Ejercicios propuestos

Ejercicio 1

Apartado a
Identificar las funciones de transferencia de los sistemas cuya salida ante entrada escalón en lazo abierto se proporciona en los archivos "P3a_sys1.mat", "P3a_sys2.mat" y "P3a_sys3.mat". Representar los sistemas en lazo abierto en Simulink y comparar las salidas obtenidas ante escalón unitario con las proporcionadas.
Nota: Los ".mat" tiene un formato de estructura con nombre *sys* con dos campos: *data* (el vector de salida) y *time* (el vector de tiempo).

Apartado b
Obtener por línea de comandos los polos y la ganancia de los tres sistemas en el dominio continuo. Comentar los resultados.

Apartado c
Dados los sistemas anteriores, obtener su error de posición (e_p) en lazo cerrado. Representar en simulink su respuesta ante entrada escalón unitario y comprobar que los errores obtenidos se corresponden con los calculados.

Apartado d
Obtener la respuesta de los sistemas en lazo cerrado ante una rampa unitaria. Comentar los resultados.

Ejercicio 2

Se sabe que un horno industrial se comporta como un sistema de primer orden. Para determinar los parámetros K y τ se lleva a cabo un experimento en el cual se le aplica una señal de entrada $r(t)=2*u(t)$ al sistema y se registran los datos de temperatura durante 10 horas con una frecuencia de muestreo de 0,01. El cambio de temperatura, y (t), se proporciona en el archivo "horno.mat".
Nota: El archivo "horno.mat" tiene un formato de estructura con nombre *horno* con dos campos: *data* (el vector de salida) y *time* (el vector de tiempo).

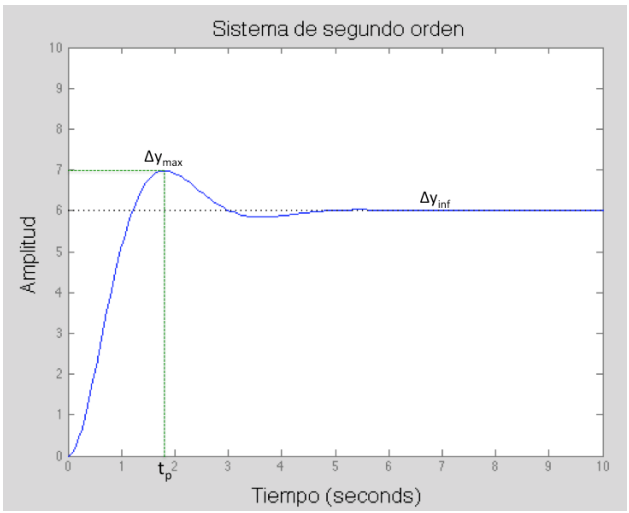
Apartado a
Determinar el valor de los parámetros K y τ.

Apartado b
Determinar el tiempo (en horas, minutos y segundos) que debe transcurrir para que y(t) alcance el estado estable. Considerar estable el sistema cuando alcanza el 95% de su valor final.

PRACTICA 1 – Análisis de sistemas en el dominio del tiempo
Sesión 3 - Sistemas de segundo orden

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

En esta sesión, se analizará el comportamiento en el dominio del tiempo de los sistemas de 2º orden. La respuesta de este tipo de sistemas presenta sobreoscilación y un periodo transitorio con oscilación.



La mayoría de los sistemas industriales se comportan como un sistema de este tipo, en el cual posteriormente el control pretende limitar parámetros como la sobreoscilación, tiempo de establecimiento y error en régimen permanente.

La función de transferencia típica de un sistema de 2º orden es la siguiente:

$$G(s) = \frac{k\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

donde:

- K es la ganancia del sistema:
  $$K = \frac{\Delta y}{\Delta u}$$
- ω_n es la frecuencia natural del sistema
- ξ es el factor de amortiguamiento

Estas dos últimas constantes pueden obtenerse de la sobreoscilación δ y el tiempo de pico t_p, obtenidos a partir de la respuesta del sistema ante una entrada escalón:

$$\delta = \frac{\Delta y_{max} - \Delta y(\infty)}{\Delta y(\infty)} = e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}}$$

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\xi^2}}$$

Ejercicios propuestos

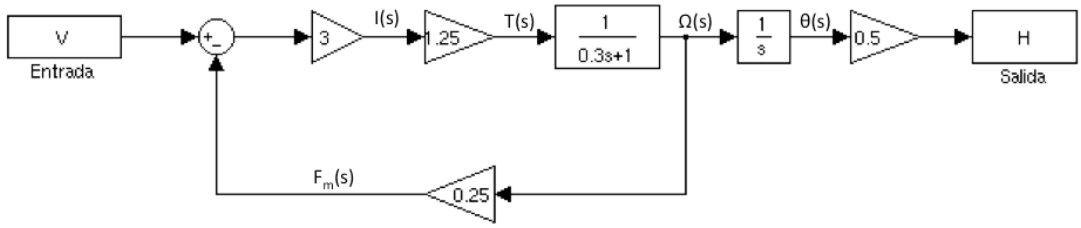
Ejercicio 1

Apartado a
Identificar las funciones de transferencia de los sistemas cuya salida ante entrada escalón en lazo abierto se proporciona en los archivos "P3b_sys1.mat" y "P3b_sys2.mat". Representar los sistemas en lazo abierto en Simulink y comparar las salidas obtenidas ante escalón unitario con las proporcionadas.
Nota: Los ".mat" tienen un formato de estructura con nombre *sys* con dos campos: *data* (el vector de salida) y *time* (el vector de tiempo).

Apartado b
Estudiar el comportamiento de los parámetros del sistema: ganancia (K), frecuencia natural del sistema (ω_n) y factor de amortiguamiento (ξ). Mostrar y comentar como varía la salida de un sistema de 2º orden ante escalón unitario en función de estos parámetros.

Ejercicio 2

Un motor eléctrico de corriente continua se acopla a una polea para levantar cajas en una planta industrial. Ante una entrada de tensión V(t) en el motor, la caja se elevará a una determinada altura h(t). El resto de variables del sistema corresponderán al ángulo girado por el motor $\theta(t)$, el par proporcionado por el motor $\tau(t)$, la intensidad que circula por el devanado del motor i(t), la fuerza electromotriz inducida en el devanado del motor $f_m(t)$ y la velocidad angular del motor $\omega(t)$. Una vez modelado el sistema se obtiene un esquema de control similar al mostrado a continuación:



Se pide:

Apartado a
Obtener la función de transferencia equivalente y mostrar la salida ante una entrada escalón con inicio en t=0. ¿Es estable el sistema? Desde un punto de vista físico, comentar el resultado.

Apartado b
Añadir un bloque multiplicador de ganancia K_G al sistema reducido y posteriormente una realimentación negativa.

Apartado c
Para un valor de K_G=20, obtener la salida del sistema ante una entrada escalón unitario con inicio en t=0.

Apartado d
¿Cuál es la ganancia en régimen permanente del sistema? ¿Qué ocurre con la ganancia si variamos el valor de K_G? ¿Qué sentido físico tiene el valor obtenido?

Apartado e
Para el valor anterior de K_G=20, calcular la sobreoscilación δ y el tiempo de pico t_p. Estudiar como afecta la variación del factor K_G en el comportamiento del sistema en régimen transitorio.

Ejercicio 3

Dado el siguiente sistema continuo de segundo orden con un cero añadido al numerador:

$$G(s)=\frac{\omega_n{}^2(1+\tau_z s)}{s^2+2\zeta\omega_n s+\omega_n{}^2}$$

Para ω_n=2 y ξ=0.5

- Evaluar el efecto de añadir un cero al numerador en s=2, s=0 y s=-2. Para ello, se debe modificar el valor de τ_Z de forma adecuada y representar la salida ante entrada escalón unitario con la función *step*.
- Dibujar los polos y los ceros en el plano complejo para cada caso y comentar el efecto que se obtiene al variar la posición del cero sobre la salida del sistema.

Para ω_n=2 y ξ=1

- Evaluar el efecto de añadir un cero al numerador en s=2, s=0, s=-0.1 y s=-2. Para ello, se debe modificar el valor de τ_Z de forma adecuada y representar la salida ante entrada escalón unitario con la función *step*.
- Dibujar los polos y los ceros en el plano complejo para cada caso y comentar el efecto que se obtiene al variar la posición del cero sobre la salida del sistema. ¿Qué diferencias hay con el sistema analizado en el apartado anterior?

Si ahora se tiene el siguiente sistema de segundo orden con un polo añadido al denominador:

$$G(s)=\frac{\omega_n{}^2}{(1+\tau_p s)(s^2+2\zeta\omega_n s+\omega_n{}^2)}$$

Para ω_n=2 y ξ=0.5

- Obtener los polos del sistema cuando τ_p=0.
- Evaluar el efecto de añadir un polo al denominador en s=-10, s=-2, s=-0.1 y s=0. Para ello, se debe modificar el valor de τ_p de forma adecuada y representar la salida ante entrada escalón unitario con la función *step*.
- Dibujar los polos y los ceros en el plano complejo para cada caso y comentar el efecto que se obtiene al variar la posición del polo sobre la salida del sistema.

PRACTICA 2 – Herramientas del análisis de sistemas continuos
Sesión 1 – Errores en régimen permanente

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

Como se ha comentado a lo largo del curso, un sistema de control lineal e invariante en el tiempo pretende que ante una señal de consigna (entrada) se produzca una salida cuyo valor en régimen permanente siga la señal de consigna. Esto ocurre siempre y cuando las señales de entrada y salida tengan la misma magnitud física. La diferencia entre la señal de salida y entrada se denomina error en régimen permanente.

El cálculo del error en régimen permanente depende de la realimentación del sistema en bucle cerrado. En este sentido, existen dos casos distintos:

Sistemas con realimentación unitaria

En este caso, la salida y la entrada son de la misma naturaleza física, por tanto:

$$E(s) = Y(s) - X(s) = \left[1 - \frac{G(s)}{1+G(s)}\right] X(s) = \left[\frac{1}{1+G(s)}\right] X(s)$$

Aplicando el teorema del valor final:

$$e_{rp} = e_{ss} = \lim_{s\to 0} s \cdot E(s) = \lim_{s\to 0} s \cdot X(s) \left[\frac{1}{1+G(s)}\right]$$

En el caso concreto de entradas conocidas se obtienen las siguientes expresiones deducidas de la ecuación anterior:

1. Entrada escalón unitario (error de posición)

$$e_p = \lim_{s\to 0} \frac{1}{1+G(s)} ; k_p = \lim_{s\to 0} G(s)$$

$$e_p = \frac{1}{1+k_p}$$

2. Entrada rampa unitaria (error de velocidad)

$$e_v = \lim_{s\to 0} \frac{1}{s(1+G(s))} ; k_v = \lim_{s\to 0} s\,G(s)$$

$$e_v = \frac{1}{k_v}$$

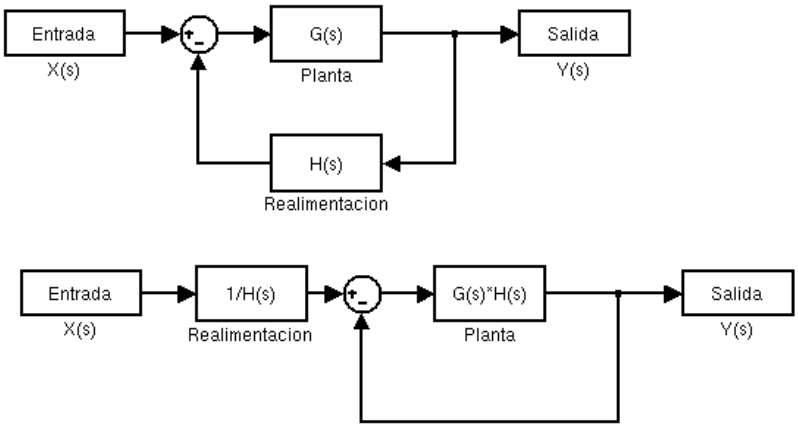
3. Entrada parábola unitaria (error de aceleración)

$$e_a = \lim_{s\to 0} \frac{1}{s^2(1+G(s))} ; k_a = \lim_{s\to 0} s^2 G(s)$$

$$e_a = \frac{1}{k_a}$$

Sistemas con realimentación no unitaria

En el caso de sistemas con realimentación no unitaria, la señal de entrada debe ser adecuada tanto en magnitud física como en rango dinámico para poder compararse con la señal de salida. Esta relación viene determinada por el bloque de realimentación H(s).



En este caso el error del sistema será el siguiente:

$$e(s) = \frac{X(s)}{H(s)} - y(s) = X(s)\frac{1}{H(s)}\left[1 - H(s)M(s)\right]$$

donde M(s) es la función de transferencia en lazo cerrado. Aplicando el teorema del valor final:

$$e_{rp} = \lim_{s\to 0} s\frac{1}{k_H}\left[1 - k_H M(s)\right]X(s)$$

donde k_H es la ganancia estática de la realimentación, es decir:

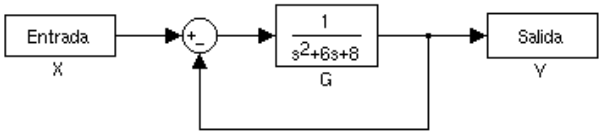
$$k_H = \lim_{s\to 0} H(s) = H(0)$$

Ejercicios propuestos

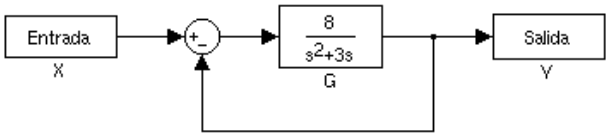
Ejercicio 1

Calcular el error de posición, velocidad y aceleración de los siguientes sistemas:

Sistema 1



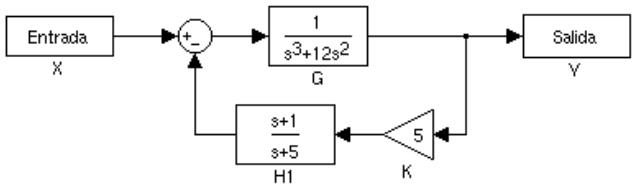
Sistema 2



Representar en Simulink y comprobar que efectivamente se obtienen los errores calculados ante entrada escalón unitario, rampa unitaria y parábola unitaria.
Nota: *Para crear una parábola unitaria se puede utilizar el bloque From Workspace*

Ejercicio 2

Calcular el error de posición, velocidad y aceleración del siguiente sistema con realimentación no unitaria:



Representar en Simulink y comprobar que efectivamente se obtienen los errores calculados ante entrada escalón unitario, rampa unitaria y parábola unitaria.
Nota: *Para crear una parábola unitaria se puede utilizar el bloque From Workspace*

PRACTICA 2 – Herramientas del análisis de sistemas continuos
Sesión 2 – Estabilidad absoluta de sistemas y lugar de las raíces

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

1. Estabilidad absoluta de sistemas

En general, un sistema dinámico se dice que es estable si tiene una respuesta acotada ante una entrada acotada. Es decir, tiene una respuesta de magnitud limitada ante una entrada o perturbación limitada.

Un sistema lineal es estable si, y solamente si, el valor absoluto de su respuesta a una entrada impulsional dentro de un intervalo finito, es finito. Es decir, la respuesta impulsional del sistema debe tender a cero cuando el tiempo aumenta. Esto se denomina la condición de convergencia absoluta.

Hay dos tipos de estabilidad:

- Absoluta: Caracteriza como es el sistema, es decir estable o inestable
- Relativa: Determina el grado de estabilidad

En esta práctica se va a analizar la estabilidad **absoluta** de sistemas utilizando Matlab.

Respuesta del sistema ante una entrada acotada

Existen diversas formas de analizar si un sistema es estable o no. La primera de ellas es analizar su respuesta ante una entrada acotada, por ejemplo, una entrada escalón. En prácticas anteriores hemos visto como modelar sistemas continuos conociendo su función de transferencia en Matlab con el comando `G=tf(n,d)`. Con la función `[Y,T] = step(G,t)` se podía obtener su respuesta ante una entrada acotada como la entrada escalón. Otra forma de obtener la respuesta del sistema ante una entrada escalón es mediante el modelado en Simulink.

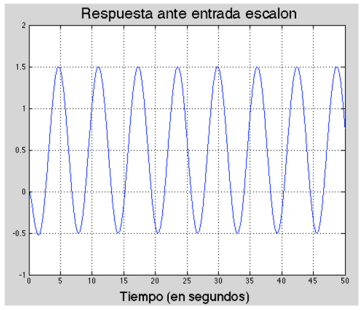
Como se ha comentado, un sistema será estable si tiene una salida acotada ante una entrada acotada. Por ejemplo, en el caso de los siguientes sistemas se obtendría:

$$G(s) = \frac{5s+20}{s^2+4s+20}$$

Estabilidad creciente
Sistema estable

$$G(s) = \frac{-2s+1}{s^3+2s^2+1s+2}$$

Estabilidad neutral
Sistema marginalmente estable



$$G(s) = \frac{200}{s(s^3+6s^2+11s+6)}$$

Estabilidad decreciente
Sistema inestable

Localización de los polos del sistema

Otra forma de analizar la estabilidad de un sistema es ver en que posición se encuentran los polos del mismo, o lo que es lo mismo, analizar las raíces de su polinomio característico que se corresponde al denominador de su función de transferencia.

En el caso de sistemas continuos, un sistema es estable si todos sus polos se encuentran en el semiplano izquierdo del eje imaginario. Para hallar los polos de un sistema continuo con Matlab se utiliza la función ya vista en sesiones anteriores `[z p k]=tf2zp(n,d)`.

Otros métodos

Cuando no es posible obtener de forma sencilla los polos del sistema porque el polinomio característico es de un grado muy elevado, existen una serie de métodos de los que se puede derivar la estabilidad absoluta de un sistema tanto continuo como discreto.

En el caso de sistemas continuos, entre otros, se puede aplicar el criterio de Routh-Hurwitz. Permite determinar el número de polos en lazo cerrado de la ecuación característica que se sitúan en el semiplano complejo derecho sin tener que factorizar el polinomio característico. Es decir, determina si existen o no raíces inestables en una ecuación polinómica sin tener que calcular sus valores.

Dada la función de transferencia:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_{n-1} s + b_n}{a_0 s^m + a_1 s^{m-1} + \cdots + a_{m-1} s + a_m}$$

donde los coeficientes son constantes reales distintos de 0 y n≤m.

La tabla de Routh quedará del siguiente modo:

$$\begin{array}{l|lll} s^m & a_0 & a_2 & a_4 \\ s^{m-1} & a_1 & a_3 & a_5 \\ s^{m-2} & b_1 & b_2 & b_3 \\ s^{m-3} & c_1 & c_2 & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ s^0 & h_1 & 0 & 0 \end{array}$$

$$b_1 = \frac{a_1 a_2 - a_0 a_3}{a_1} \qquad b_2 = \frac{a_1 a_4 - a_0 a_5}{a_1} \qquad b_3 = \frac{a_1 a_6 - a_0 a_7}{a_1}$$

$$c_1 = \frac{b_1 a_3 - a_1 b_2}{b_1} \qquad c_2 = \frac{b_1 a_5 - a_1 b_3}{b_1} \qquad c_3 = \frac{b_1 a_7 - a_1 b_4}{b_1}$$

- Si todos los valores de la primera columna son positivos, entonces todas las raíces del polinomio serán positivas y consecuentemente todos los polos estarán en el semiplano complejo izquierdo y el sistema será **estable**.
- Si por el contrario se producen cambios de signo, entonces es posible asegurar que habrá tantas raíces en el semiplano complejo derecho como cambios de signo se observen en la primera columna y el sistema será **inestable**.

Existen, no obstante, casos especiales en los que hay que aplicar alguna aproximación para poder completar la tabla de Routh:

1. Existe un 0 en la primera columna, lo que impide calcular el siguiente término al ser infinito, al dividirse por 0.

Solución: Sustituir el término que vale 0 con un número positivo muy pequeño que tiende a cero (ε). Y entonces, se continúa rellenando la tabla teniendo en cuenta es número ε. Los cambios de signo se evalúan aproximando ε a un número muy pequeño.

2. Existe una fila de 0s.

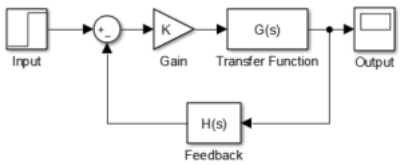
Solución: Sustituir toda la fila de 0 por la que se obtendría de derivar el polinomio correspondiente a la fila anterior a la de ceros, y después continuar operando como en el caso base.

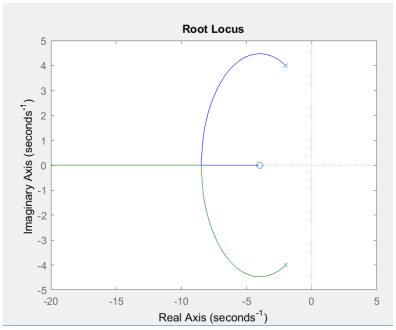
2. Lugar de las raíces

El comportamiento dinámico de un sistema de control viene dado por la posición de los polos y de los ceros en el plano complejo. Un modo de estudiar la posición de éstos es mediante la técnica del lugar de las raíces. En esta parte práctica, el objetivo es familiarizarse con las herramientas que permiten dibujar el lugar de las raíces de sistemas de control, así como afianzar conceptos vistos en clase de teoría en relación a esta técnica.

El lugar de las raíces permite observar la evolución de las raíces de la ecuación característica (polos del sistema en lazo cerrado), al variar K.

Para construir el lugar de las raíces, es necesario trabajar con la función de transferencia en lazo abierto con la forma K(num/den). Utilizando la orden *rlocus(num,den)* se genera el lugar de las raíces de un sistema de una entrada y una salida, donde num y den representan dos vectores de igual dimensión con los coeficientes de la ecuación polinómica de numerador y denominador de la función de transferencia del sistema en lazo abierto. La ganancia K es obtenida automáticamente si el comando se utiliza con la sintaxis *[r,K]=rlocus(num,den)* donde r son las raíces y K todos los valores de ganancia que se obtienen para la evolución de los polos en lazo cerrado.





Si se conoce un vector de ganancia K suministrado por el usuario, es decir todos los valores de ganancia para los cuales los polos en lazo cerrado son calculados, se puede emplear el comando con la sintaxis *[r,K]=rlocus(num,den,K)* donde r son las raíces y K todos los valores de ganancia que se obtienen para la evolución de los polos en lazo cerrado. Este comando es especialmente útil cuando se producen errores en el cálculo aproximado que hace MATLAB del lugar de las raíces. Supongamos que los valores de ganancia se mueven entre [0 200], entonces puede interesar no dejar a MATLAB que calcule K aplicando un paso automático. En este caso, uno definiría su propia manera de iterar K entre ambos valores. Supongamos que se quiere definir un paso de iteración de por ejemplo 0.5, entonces se emplearía la instrucción K=[0:0.5:200]. También, puede darse el caso que dentro de ese rango se necesite aplicar dos intervalos distintos (por ejemplo [0 10] y [10 200]) con pasos distintos (por ejemplo 0.1 y 0.5, respectivamente), entonces se podrían definir ambos intervalos de la siguiente forma K1=0:0.1:10 y K2=10:0.5:200, y después definir K como K=[K1 K2]. Posteriormente ya se podría llamar a *[r,K]=rlocus(num,den,K)* para aplicar los intervalos definidos.

Por otro lado, la orden *[K,p]=rlocfind(num,den)* permite obtener el valor de K y el valor numérico de las raíces en un punto específico del lugar de las raíces. En este caso, es posible controlar un cursor con forma de cruz para seleccionar el punto del lugar de las raíces que se desea.

Ejercicios propuestos

Ejercicio 1

Dados los sistemas continuos definidos por las siguientes funciones de transferencia:

$$G(s) = \frac{s-4}{s^5+s^4+3s^3+9s^2+16s+10}$$

$$G(s) = \frac{s+8}{s^4+s^3+2s^2+2s+3}$$

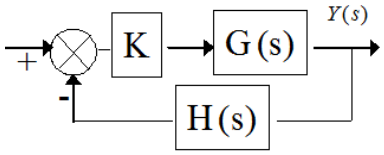
$$G(s) = \frac{s^2+3s+8}{s^5+4s^4+8s^3+8s^2+7s+4}$$

Apartado a
Estudiar su estabilidad absoluta analizando su comportamiento ante una entrada escalón.

Apartado b
Estudiar su estabilidad absoluta analizando la posición de los polos del sistema (raíces del polinomio característico). Utilizar la función *tf2zp* o la función *roots*.

Ejercicio 2

Dado el esquema de la figura, realiza un script en MATLAB que calcule el lugar de las raíces, así como los polos y ceros de la ecuación característica de los siguientes cuatro sistemas, y muestre dicha información en cuatro gráficas dentro de una misma figura.



a)	**b)**
G(s)= $(s^2+1)/s(s+2)$ H(s)=1	G(s)= $(s+2)/(s^2+2s+1)$ H(s)=1
c)	**d)**
G(s)= $1/s(s^2+4s+5)$ H(s)=1	G(s)= $s(s+5)/(s+2)(s+5)(s^2+4s+7)$ H(s)=1

Ejercicio 3

Para los sistemas del ejercicio anterior, se pide calcular, con las gráficas del lugar de las raíces dibujadas y las instrucciones de MATLAB adecuadas, los siguientes datos:

- Valores de K para los que el sistema es inestable en cada caso, si los hubiera.

- Coordenadas de los puntos de ruptura y de dispersión, así como valor de K para ellos, si los hubiera.

- Número de asíntotas y grados de éstas.

Ejercicio Opcional

Apartado a
Realizar un script de Matlab (*routh.m*) que devuelva la tabla de Routh-Hurwitz correspondiente al polinomio característico introducido como parámetro.

Apartado b
Mejorar el script para que tenga en cuenta los casos especiales

Apartado c
Mejorar el script para que determine además si el sistema es estable o no y lo muestre por pantalla mediante un mensaje.

Ejemplo de ejecución de la función (para apartados a y b):

```
Polinomio característico: s^3+2*s^2+3*s+1

>>syms EPS
>>ra=routh([1 2 3 1],EPS)
       ra =

[   1, 3]
[   2, 1]
[ 5/2, 0]
[   1, 0]

Polinomio característico: s^4+s^3+2*s^2+2*s+3

>>syms EPS
>>ra=routh[1 1 2 2 3],EPS)
ra =

[               1, 2, 3]
[               1, 2, 0]
[             EPS, 3, 0]
[ (2*EPS - 3)/EPS, 0, 0]
[               3, 0, 0]
```

PRACTICA 2 – Herramientas del análisis de sistemas continuos
Sesión 3 – Ejercicio adicional

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

Ejercicio

Dados los siguientes sistemas continuos:

$$G1(s) = \frac{5s - 15}{s^3 + s^2 + 8s - 60}$$

$$G2(s) = \frac{5}{s^2 + 4s + 20}$$

$$G3(s) = \frac{3s - 6}{s^3 + 7s^2 + 19s - 13}$$

Apartado 1
Obtener su salida ante entrada escalón unitario durante 10 segundos y la representación del lugar de las raíces. Muestra estas gráficas en una única figura de dimensiones 3x2 con la función *subplot*.

Apartado 2
Varía ahora la duración de la representación a 20 segundos. A la vista de los resultados comentar la diferencias y semejanzas entre los diferentes sistemas continuos propuestos.

Apartado 3
Reducir los sistemas a un orden inferior siempre que sea posible. Representar de nuevo lo pedido en al Apartado 1 y comentar los resultados.

Apartado 4
Desplazar el polo inestable del sistema 1 en dirección al eje imaginario y mostrar por pantalla la salida del sistema junto con el lugar de las raíces para distintos valores. Comentar qué efecto tiene sobre la salida del sistema.

PRACTICA 2 – Herramientas del análisis de sistemas continuos
Sesión 4 – Diagramas de Bode: Comportamiento en frecuencias

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

Construir el diagrama de Bode en MATLAB

Los diagramas de Bode indican el margen de ganancia, el margen de fase, la ganancia en continuo, el ancho de banda y otros parámetros de interés.

Para construir un diagrama de Bode y que se dibuje, es posible emplear la orden `bode(num,den)`. Tanto en una orden como en otra `num` y `den` representan dos vectores de igual dimensión con los coeficientes de la ecuación polinómica de numerador y denominador de la función de transferencia del sistema en lazo abierto. Los coeficientes del polinomio `num` están operados con la ganancia K.

Para construir el diagrama de Bode también se emplea la orden `[mag, fase, w]=bode(num,den,wu)`. Ésta genera el diagrama de bode y devuelve la respuesta en frecuencias del sistema en las matrices de magnitud, fase y frecuencia, w. Esta orden no dibuja gráficamente el diagrama sobre la pantalla. Las matrices de magnitud y fase contienen la amplitud o módulo y los ángulos de fase de la respuesta en frecuencia del sistema evaluado en los puntos de frecuencia indicados en la wu que el usuario pasa como parámetro. El ángulo de fase se devuelve en grados, no obstante, la magnitud es necesario convertirla a decibelios en algunas versiones de MATLAB mediante la expresión `magdB=20*log 10(mag)`.

Si se desea especificar el rango de magnitud para que se encuentre, entre dos valores específicos, por ejemplo -45dB y 45dB es posible introducir líneas no visibles en -45dB y +45dB con el siguiente ejemplo: `dBmax=45*ones(1,100); dBmin=-45*ones(1,100);` Después, se pueden dibujar con la orden `semilogx(w,magdB,'o',w,magdB,'-`',w,dBmax,'--i',w,dBmin,':i')` donde el número de puntos de `dBmax` y `dBmin` deben coincidir con el número de puntos de la frecuencia `w`. El resultado será una curva de magnitud `magdB` con las marcas 'o' como elemento de píncel. El parámetro 'i' de la expresión hace que ciertas líneas no se dibujen o lo hagan con una tinta invisible. Si la expresión se modifica por `semilogx(w,magdB,'o',w,magdB,'-`',w,dBmax,'--',w,dBmin,':')` entonces las líneas de +45dB y -45dB se harán visibles en la pantalla. Los rangos para la magnitud, normalmente, son múltiplos de 5dB, 10dB, 20dB o 50dB, aunque existen excepciones.

Si se desea especificar el rango de ángulo de fase, se procede de modo similar a como se hizo para la magnitud. Por ejemplo para que se encuentre, entre dos valores específicos, tales como -145º y 115º es posible introducir líneas no visibles en esos límites con el siguiente ejemplo: `fmax=115*ones(1,100); fmin=-145*ones(1,100);` Después, se pueden dibujar con la orden `semilogx(w,fase,'o',w,fase,'-`',w,fmax,'--i',w,fmin,':i')` donde el número de puntos de `fmax` y `fmin` deben coincidir con el número de puntos de la frecuencia `w`. El resultado será una curva de fase `fase` con las marcas 'o' como elemento de píncel. El parámetro 'i' de la expresión hace que ciertas líneas no se dibujen o lo hagan con una tinta invisible. Si la expresión se modifica como se ha hecho antes entonces las líneas de +115º y -145º se harán visibles en la pantalla. Los rangos para la fase, normalmente, son múltiplos de 5º, 10º, 50º o 100º, aunque también existen excepciones como para el caso de las magnitudes.

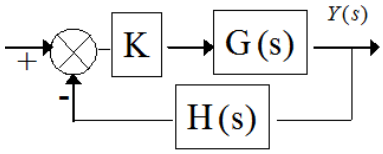
Si se quiere dibujar el diagrama de Bode construido con la orden `[mag, fase, w]=bode(num,den,wu)`, es necesario añadir después la orden `logspace`. Ésta, además permite especificar el rango de frecuencias que se quieren mostrar. La sintaxis de dicha expresión puede ser de dos tipos: `logspace(d1,d2)` o `logspace(d1,d2,n)`. La primera

expresión genera un vector de 50 puntos espaciados logarítmicamente por igual entre las décadas 10^{d1} y 10^{d2}. La segunda de las expresiones genera n puntos espaciados logarítmicamente por igual entre las décadas 10^{d1} y 10^{d2}. Por ejemplo, para generar 100 puntos entre una frecuencia de 1 rad/seg y otra frecuencia de 1000 rad/s, se introducirá la orden `logspace(0,3,100)`.

Ejercicios propuestos

Ejercicio 1

Dado el esquema de la figura, realiza un script en MATLAB que calcule el diagrama de Bode en los siguientes casos.



a)	**b)**
G(s)= 1/ (s^2+4s+25) H(s)=1; K=25	G(s)=(s^2+0.2s+1)/s(s^2+1.2s+9) H(s)=1; K=9
c)	**d)**
G(s)= (s+1)/(s+2)(s+5) H(s)=1; K=10	G(s)=(s+0.5)/ (s^3+ s^2+1) H(s)=1; K=1

Ejercicio 2

Dado el siguiente sistema:

$$G(s) = \frac{20(s+1)}{(s+5)(s^2+2s+4)}$$

Dibuja el diagrama de Bode con MATLAB y después analiza la información que se obtiene de él sin llevar a cabo cálculos numéricos.

Se pide:

a) Indica los valores de las frecuencias de corte del diagrama de Bode asintótico, a partir de la función de transferencia G(s) normalizada del sistema
b) A la vista del diagrama de Bode calculado por Matlab, determina el valor inicial y final del diagrama de amplitudes y del diagrama de fases
c) A la vista del diagrama de Bode calculado por Matlab, determina la frecuencia de resonancia del sistema y calcular el valor en magnitud para una década superior y para una octava inferior

   Frecuencia de resonancia:

   $\omega_r = \omega_n\sqrt{1-2\zeta^2}$
d) ¿Entre que dos valores de frecuencia se observa principalmente el efecto del cero sobre la respuesta en frecuencia del sistema?
e) Calcula el margen de fase y el margen de ganancia del sistema. A partir de los resultados obtenidos, determina la estabilidad del sistema en bucle cerrado.

PRACTICA 3 – Modelado y análisis de sistemas discretos

Profesores:
Teoría: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es
José Luis Ramón Carretero, jl.ramon@ua.es

En esta práctica, se analizarán las herramientas de análisis de sistemas discretos en Matlab y Simulink. La mayoría de estas herramientas son similares a las que se utilizan para sistemas continuos, desde la generación de funciones de transferencia hasta el modelado de sistemas discretos en Simulink.

3.1 Funciones de transferencia. Sistemas discretos.

En el caso de trabajar en el dominio discreto se puede obtener la función de transferencia con la función `G = tf(num,den,T)`, donde:

- num: vector de coeficientes del numerador
- den: vector de coeficientes del denominador
- T: periodo de muestreo, si indicamos -1 se toma un período de muestreo por defecto (1 segundo)

De forma parecida a los sistemas continuos, se pueden obtener los ceros, polos y ganancia de un sistema discreto mediante:

```
[z,p,k] = tf2zpk(n,d)
```

También es posible hacer la operación inversa, es decir, obtener la función de transferencia a partir de los ceros, polos y ganancia del sistema mediante la función `G = zpk(z,p,k,T)`, donde:

- ceros: vector con los ceros de la función de transferencia
- polos: vector con los polos
- ganancia: es un valor que multiplica a la función de transferencia
- T: periodo de muestreo, si indicamos -1 se toma un período de muestreo por defecto (1 segundo)

También es posible pasar de un dominio a otro. Por ejemplo, si se quiere pasar de un sistema continuo a un sistema discreto se utiliza:

```
[Nz,Dz] = c2dm (N,D,Ts,'metodo')
```

Esta función permite la discretización de un modelo en tiempo continuo, cuya función de transferencia viene dada por los polinomios numerador y denominador. Como tercer parámetro se especifica el periodo de muestreo. El último parámetro proporciona una cadena de caracteres que indica el método con el que se va a hacer la discretización. Hay varias posibilidades:

- 'zoh': Discretización utilizando mantenedor de orden cero (ZOH). Es la opción
- por defecto.
- 'foh': Discretización utilizando mantenedor de orden uno (FOH).
- 'tustin': Discretización mediante aproximación trapezoidal.
- 'prewarp': Discretización trapezoidal con prewarping.
- 'matched': Discretización mediante emparejamiento de polos y ceros

Si se desea hacer la operación contraria: transformar un sistema discreto en uno continuo, se puede utilizar la siguiente función:

```
[N,D] = d2cm(Nz,Dz,Ts,'metodo')
```

3.2 Representación de sistemas discretos

Para la representación de sistemas discretos en Matlab es necesario usar la función `stem` en lugar de la función `plot`, que permite representar valores discretos en una gráfica.

```
>> Ts=0.1; % período de muestreo
>> [nz,dz] = c2dm (n,d,Ts,'zoh');

>> t=10 % tiempo de representación
>> G = tf(nz,dz,Ts);
>> [Y,T] = step(G,t); % respuesta ante escalón

>> figure
>> stem(T,Y)
>> grid
>> title('Respuesta ante entrada escalón')
>> xlabel('Tiempo(en segundos)')
```

El resultado mostrado será el siguiente:

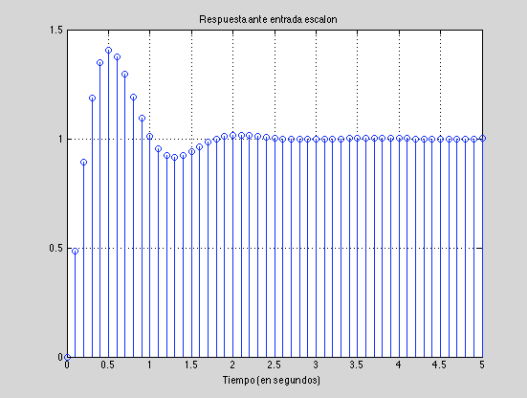


***Figura 4**. Respuesta ante entrada escalón de un sistema discreto*

3.3. Sistemas discretos en Simulink

Al igual que los sistemas continuos, los sistemas discretos se pueden representar en Simulink mediante su función de transferencia. Para ello, existen dos posibilidades:

1. Utilizar el bloque Discrete -> Discrete Transfer Function, donde los coeficientes de numerador y denominador se introducen como vectores, al igual que en el caso de los sistemas continuos. Además hay que especificar el período de muestreo.

2. Utilizar el bloque Discrete -> Discrete Filter, donde los coeficientes de numerador y denominador se introducen como vectores, al igual que en el caso anterior, pero con la particularidad de que se trabaja en potencias negativas de z.

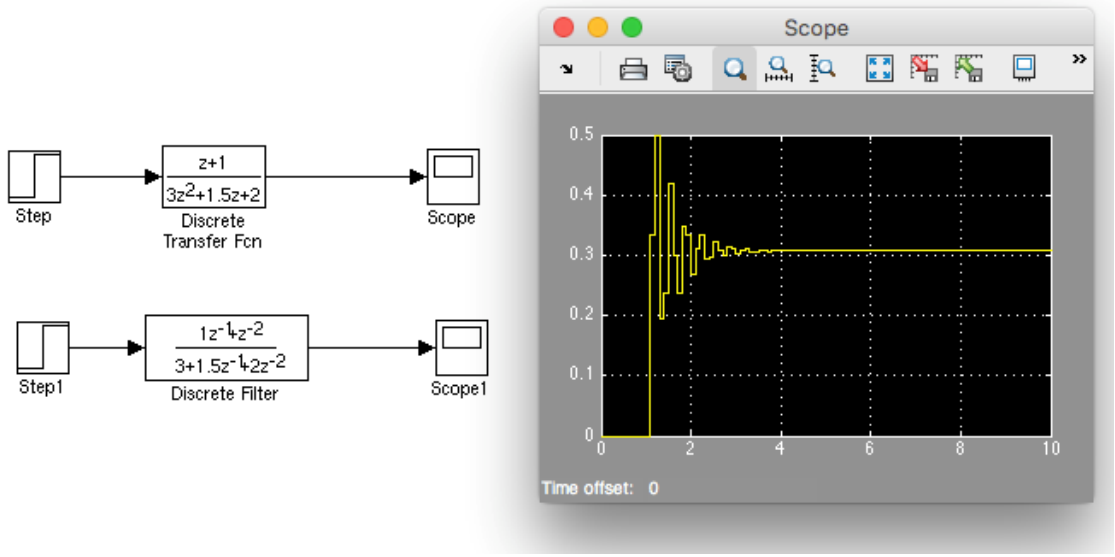
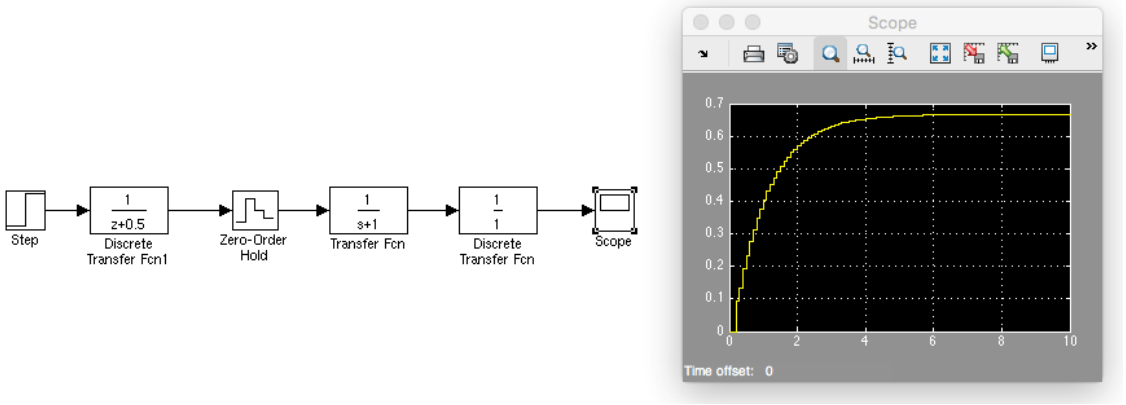


Figura 7. Representación de sistemas discretos

En ocasiones, es necesario combinar bloques en el dominio continuo y en el dominio discreto (sistemas mixtos). Para ello, se pueden emplear bloqueadores y muestreadores. En SIMULINK, un bloqueador se puede introducir mediante el bloque *Discrete -> Zero-Order Hold* o mediante el bloque *Discrete -> First-Order Hold* según el orden del mismo. Un muestreador básico es equivalente a usar el bloque *Discrete -> Discrete Transfer Function* con el numerador y el denominador a 1.



Ejercicios propuestos

Ejercicio 1

Dados los siguientes sistemas continuos:

a)

$$G(s) = \frac{5s + 20}{s^2 + 4s + 20}$$

b)

$$G(s) = \frac{-2s + 1}{s^3 + 2s^2 + 1s + 2}$$

c)

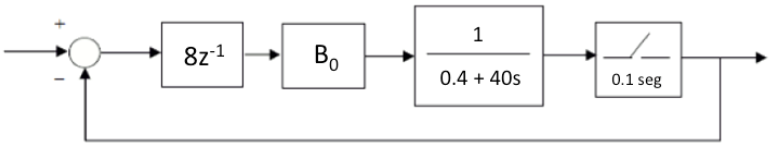
$$G(s) = \frac{200}{s(s^3 + 6s^2 + 11s + 6)}$$

Se pide:

1. Discretizar los sistemas (T_m=0.1) con un bloqueador de orden 0 y uno de orden 1.
2. Obtener sus polos y ceros en el plano z.
3. Analizar la estabilidad de los sistemas propuestos en el domino discreto.

Ejercicio 2

Dado el siguiente sistema con T_m=0.1:



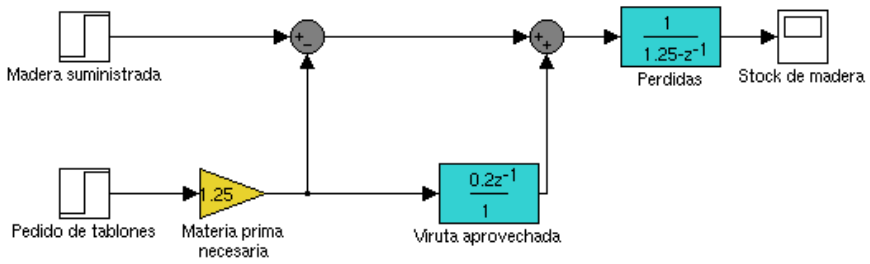
Se pide:

1. Calcular la función de transferencia del sistema en bucle cerrado en el dominio continuo (utilizar el método 'foh' para convertir la función discreta en continua)
2. Calcular la función de transferencia del sistema en bucle cerrado en el dominio discreto (utilizar el método 'foh' para convertir la función continua en discreta)
3. Obtener la respuesta del sistema ante entrada escalón para un tiempo de simulación de 10 segundos. ¿Es posible saber si el sistema es estable?
4. Obtener la respuesta del sistema para un tiempo de simulación de 50 segundos y comentar.

Ejercicio 3

En este ejercicio se va a simular el funcionamiento de una fábrica de tablones. Supongamos que a la fábrica llega a diario un suministro de madera sin tratar (materia prima) de la cuál un 80% se convierte en tablones y el restante 20% se convierte en viruta, que tratada adecuadamente se puede transformar en conglomerado que se puede reutilizar para hacer tablones. Este proceso dura un día. Además, del stock de materia prima, un 20% de la madera se pudre y se pierde.

Si modelamos el sistema con dos entradas: el suministro de madera sin tratar y el número de tablones que se pide a la fábrica; y definimos como salida el comportamiento del stock de madera de la fábrica, podemos simular el sistema del siguiente modo:



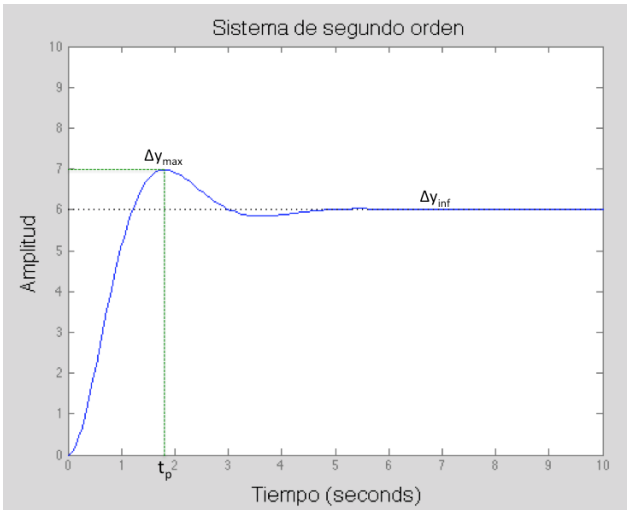
Se pide:

1. Introducir el esquema Simulink propuesto (utilizar un período de muestreo de 1 día para todos los bloques).
2. Simular el comportamiento del sistema para las siguientes condiciones:
    a. Incremento de 30 unidades/día de madera sin tratar suministrada y pedido de tablones constante, es decir incremento de 0 unidades/día
    b. Incremento de 30 unidades/día de madera sin tratar y pedido de tablones equivalente, es decir, incremento de 30 unidades/día
    c. Suministro de madera sin tratar constante (0 unidades/día) e incremento de 20 unidades/día en el pedido de tablones.
3. Guardar los resultados obtenidos mediante el bloque To Workspace y representarlos mediante la función Stem.
4. ¿Cómo afecta la variación del porcentaje de pérdidas y de aprovechamiento de viruta en el comportamiento del sistema?

PRACTICA 3 – Análisis de sistemas en el dominio del tiempo
Sesión 2 - Sistemas de segundo orden

Profesores:
Teoría y prácticas: Pablo Gil Vázquez, pablo.gil@ua.es
Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es

En esta práctica, se analizará el comportamiento en el dominio del tiempo de los sistemas de 2º orden. La respuesta de este tipo de sistemas presenta sobreoscilación y un periodo transitorio con oscilación.



La mayoría de los sistemas industriales se comportan como un sistema de este tipo, en el cual posteriormente el control pretende limitar parámetros como la sobreoscilación, tiempo de establecimiento y error en régimen permanente.

La función de transferencia típica de un sistema de 2º orden es la siguiente:

$$G(s) = \frac{k\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

donde:

- K es la ganancia del sistema:
  $$K = \frac{\Delta y}{\Delta u}$$
- ω_n es la frecuencia natural del sistema
- ξ es el factor de amortiguamiento

Estas dos últimas constantes pueden obtenerse de la sobreoscilación δ y el tiempo de pico t_p, obtenidos a partir de la respuesta del sistema ante una entrada escalón:

$$\delta = \frac{\Delta y_{\max} - \Delta y(\infty)}{\Delta y(\infty)} = e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}}$$

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\xi^2}}$$

Ejercicios propuestos

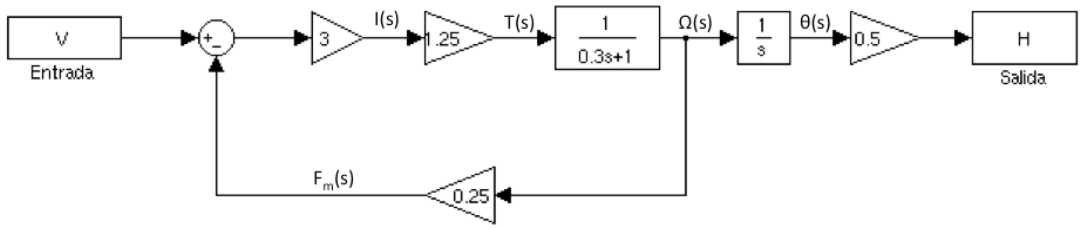
Ejercicio 1

a) Identificar las funciones de transferencia de los sistemas cuya salida ante entrada escalón en lazo abierto se proporciona en los archivos "P3b_sys1.mat" y "P3b_sys2.mat. Representar los sistemas en lazo abierto en Simulink y comparar las salidas obtenidas ante escalón unitario con las proporcionadas.
Nota: Los ".mat" tienen un formato de estructura con nombre *sys* con dos campos: *data* (el vector de salida) y *time* (el vector de tiempo).

b) Estudiar el comportamiento de los parámetros del sistema: ganancia (K), frecuencia natural del sistema (ω_n) y factor de amortiguamiento (ξ). Mostrar y comentar como varía la salida de un sistema de 2º orden ante escalón unitario en función de estos parámetros.

Ejercicio 2

Un motor eléctrico de corriente continua se acopla a una polea para levantar cajas en una planta industrial. Ante una entrada de tensión V(t) en el motor, la caja se elevará a una determinada altura h(t). El resto de variables del sistema corresponderán al ángulo girado por el motor $\theta(t)$, el par proporcionado por el motor $\tau(t)$, la intensidad que circula por el devanado del motor i(t), la fuerza electromotriz inducida en el devanado del motor $f_m(t)$ y la velocidad angular del motor $\omega(t)$. Una vez modelado el sistema se obtiene un esquema de control similar al mostrado a continuación:



Se pide:

a) Obtener la función de transferencia equivalente y mostrar la salida ante una entrada escalón con inicio en t=0. ¿Es estable el sistema? Desde un punto de vista físico, comentar el resultado.
b) Añadir un bloque multiplicador de ganancia K al sistema reducido y posteriormente una realimentación negativa.
c) Para un valor de K=20, obtener la salida del sistema ante una entrada escalón unitario con inicio en t=0.
d) ¿Cuál es la ganancia en régimen permanente del sistema? ¿Qué ocurre con la ganancia si variamos el valor de K? ¿Qué sentido físico tiene el valor obtenido?
e) Para el valor anterior de K=20, calcular la sobreoscilación δ y el tiempo de pico t_p. Estudiar como afecta la variación del factor K en el comportamiento del sistema en régimen transitorio.

Ejercicio 3

Dado el siguiente sistema continuo de segundo orden con un cero añadido al numerador:

$$G(s)=\frac{\omega_n{}^2(1+\tau_z s)}{s^2+2\zeta\omega_n s+\omega_n{}^2}$$

Para ω_n=2 y ξ=0.5

- Evaluar el efecto de añadir un cero al numerador en s=2, s=0 y s=-2. Para ello, se debe modificar el valor de τ_Z de forma adecuada y representar la salida ante entrada escalón unitario con la función *step*.
- Dibujar los polos y los ceros en el plano complejo para cada caso y comentar el efecto que se obtiene al variar la posición del cero sobre la salida del sistema.
- Transformar el sistema al domino discreto para cada caso, obtener la posición de los polos y ceros y dibujar junto con el círculo unidad. Comentar como varía la posición del cero en su representación discreta.

Para ω_n=2 y ξ=1

- Evaluar el efecto de añadir un cero al numerador en s=2, s=0, s=-0.1 y s=-2. Para ello, se debe modificar el valor de τ_Z de forma adecuada y representar la salida ante entrada escalón unitario con la función *step*.
- Dibujar los polos y los ceros en el plano complejo para cada caso y comentar el efecto que se obtiene al variar la posición del cero sobre la salida del sistema. ¿Qué diferencias hay con el sistema analizado en el apartado anterior?

Si ahora se tiene el siguiente sistema de segundo orden con un polo añadido al denominador:

$$G(s)=\frac{\omega_n{}^2}{(1+\tau_p s)(s^2+2\zeta\omega_n s+\omega_n{}^2)}$$

Para ω_n=2 y ξ=0.5

- Obtener los polos del sistema cuando τ_p=0.
- Evaluar el efecto de añadir un polo al denominador en s=-10, s=-2, s=-0.1 y s=0. Para ello, se debe modificar el valor de τ_p de forma adecuada y representar la salida ante entrada escalón unitario con la función *step*.
- Dibujar los polos y los ceros en el plano complejo para cada caso y comentar el efecto que se obtiene al variar la posición del polo sobre la salida del sistema.
- Transformar el sistema al domino discreto para cada caso, obtener la posición de los polos y ceros y dibujar junto con el círculo unidad. Comentar como varía la posición de los polos en su representación discreta.

PRACTICA 4. Modelado e identificación de un servomotor

Profesores:

Teoría: Pablo Gil Vázquez, pablo.gil@ua.es

Prácticas: Andrés Úbeda Castellanos, andres.ubeda@ua.es

José Luis Ramón Carretero, jl.ramon@ua.es

En esta práctica, se realizará el modelado de un sistema real para obtener su función de transferencia, y posteriormente se realizará un breve análisis de su comportamiento. El modelo de un motor de corriente continua se puede obtener analíticamente mediante la aplicación de principios mecánicos y eléctricos de funcionamiento, como ya fue visto en la transparencia 18 del tema 1- sesión 2, y que aquí se repasa, viendo su aplicación a un motor algo más complejo, que dispone de un tren de engranaje ensamblado en el eje de giro del motor.

Sesión 1: Modelado del sistema analíticamente

S.1.1.Modelado de un motor de corriente continua sin tren de engranajes

La **parte eléctrica** del motor viene modelada por la ecuación del circuito inducido resuelta por las leyes de *Kirchoff*, donde se relaciona la tensión de entrada $v_i(t)$ con el circuito los componentes del circuito eléctrico.

$$v_i(t) = R_i i_i(t) + L_i \frac{di_i(t)}{dt} + v_m(t) = R_i i_i(t) + L_i \frac{di_i(t)}{dt} + K_b \frac{d\theta(t)}{dt} \quad (1)$$

donde R_i es la resistencia, L_i el valor de la inductancia y donde $v_m(t) = K_b \frac{d\theta(t)}{dt} = K_b \omega_m(t)$ es la tensión de la fuerza electromotriz, la cual es dependiente de la velocidad de giro del eje del motor $\omega_m(t)$, y de una constante de proporcionalidad llamada constante de fuerza electromotriz K_b.

En el caso de que la inductancia del motor L_i sea mucho más pequeña que el valor de la resistencia R_i , es posible despreciar este valor, en cuyo caso la ecuación resultante quedaría simplificada y de la forma:

$$v_i(t) = R_i i_i(t) + v_m(t) = R_i i_i(t) + K_b \omega_m(t) \quad (2)$$

Y la **parte mecánica** viene modelada por la ecuación que relaciona inercias y amortiguamiento viscoso, que se obtiene por aplicación de segunda ley de Newton, y donde se relaciona el par aplicado de entrada $T_m(t)$ con los componentes mecánicos.

$$T_m(t) = J_m \frac{d^2\theta_m(t)}{dt} + B_m \frac{d\theta_m(t)}{dt} = J_m \frac{d\omega_m(t)}{dt} + B\omega_m(t) \quad (3)$$

donde J_m es el momento de inercia de la carga del motor, B_m el rozamiento viscoso en el eje del motor (se supone que la carga está en el eje y coinciden) y donde $\omega_m(t) = \frac{d\theta_m(t)}{dt}$ es la velocidad de giro del eje del motor.

Además, la relación entre la parte eléctrica y continua en un motor de corriente continua viene dada por la ecuación:

$$T_m(t) = n_m K_t i_i(t) \quad (4)$$

Donde el par del motor, viene determinado por la intensidad del circuito eléctrico multiplicada por una constante de proporcional, conocida por constante del par motor, y que se denota por K_t, y por un parámetro conocido como la eficiencia del motor n_m que en condiciones ideales valdrá 1.

S.1.2.Modelado de un motor de corriente continua añadiendo un tren de engranajes

El tren de engranaje permite modificar la relación par-velocidad entre elemento del motor y la carga que soporta. Así la relación par-velocidad se modifica proporcionalmente a la relación de transmisión de los dos elementos mecánicos acoplados de los que consta el tren de engranaje $K_{gi} = \frac{N_1}{N_2}$.

Si además se acoplan en cascada un segundo tren de engranaje al primero, la relación de par-velocidad se modifica proporcionalmente a $K_{ge} \cdot K_{gi} = \frac{N_4}{N_3} \cdot \frac{N_2}{N_1}$ donde $K_{ge} = \frac{N_4}{N_2}$ es el segundo tren de engranaje.

En el servomotor SRV02 que hay identificar y con el que hay que trabajar en la práctica, el eje del motor tiene acoplado dos trenes de engranaje de 2 elementos cada uno, por lo que la relación de engranajes se puede poner como:

$$K_g = K_{ge} \cdot K_{gi} = \frac{N_4}{N_3} \cdot \frac{N_2}{N_1}$$

Al añadir un tren de engranajes, la parte eléctrica del motor no se ve modificada pero si lo hace la parte mecánica, que como se ha observado modifica el par de fuerza a la salida y por lo tanto, el desplazamiento en la carga es proporcional al desplazamiento en el eje de giro del motor y la velocidad de giro también es proporcional.

$$\theta_m(t) = K_g \theta_l(t) \text{ y } \omega_m(t) = K_g \omega_l(t)$$

Y del mismo modo, el par resultante a la salida (en el eje de la carga) viene dado por:

$$T_l(t) = n_g K_g T_m(t)$$

donde n_g es la eficiencia del sistema de engranaje, sino hay perdida, pues sólo depende del par del motor T_m y de la relación de engranaje K_g.

Pero además, el tren de engranaje, añade un desplazamiento de la carga con respecto al eje de giro. Es decir, la carga ya no apoyará directamente sobre el eje de giro, por lo que habrá que modificar ligeramente el modelo, para contemplar este hecho. Así, la ecuación (3), al incorporar el tren de engranaje con el desplazamiento de carga queda como:

$$T_l(t) = (n_g {K_g}^2 J_m + J_l) \frac{d\omega_l(t)}{dt} + (n_g {K_g}^2 B_m + B_l)\omega_l(t) = n_g K_g T_m(t) \quad (5)$$

S.1.3.Modelado del MOTOR de corriente continua real: Caso práctico SVR02 de QUANSEC

El modelo de un motor de corriente continua se compone de una ecuación diferencial fruto de la combinación de las ecuaciones de la parte eléctrica y de la parte mecánica. Para el caso que nos ocupa, si se combinan las ecuaciones (2) y (4) de la parte eléctrica se tiene que:

$$T_m(t) = n_m K_t i_i(t) = n_m K_t \left(\frac{v_i(t) - K_b \omega_m(t)}{R_i}\right)$$

Y si esta nueva ecuación se combina con la ecuación (5) que define la parte mecánica, y se sabe que $\omega_m(t) = K_g \omega_l(t)$ entonces, se tiene que:

$$\left(n_g {K_g}^2 J_m + J_l\right) \frac{d\omega_l(t)}{dt} + \left(n_g {K_g}^2 B_m + B_l\right)\omega_l(t) = n_g K_g n_m K_t \left(\frac{v_i(t) - K_b K_g \omega_l(t)}{R_i}\right)$$

Después agrupando términos, y considerando que la velocidad angular en la carga $\omega_l(t)$ es la salida del sistema, y que la entrada es la tensión que se aplica a la entrada del circuito del motor $v_i(t)$, entonces se tiene la ecuación diferencial que modela el motor de la práctica, ecuación (6).

$$\left(n_g {K_g}^2 J_m + J_l\right) \frac{d\omega_l(t)}{dt} + \left[\left(n_g {K_g}^2 B_m + B_l\right) - \frac{n_g {K_g}^2 n_m K_t K_b}{R_i}\right] \omega_l(t) = \left(\frac{n_g K_g n_m K_t}{R_i}\right) v_i(t) \quad (6)$$

Ejercicio 1 (Sesión 1)

(a) Obtener la función de transferencia del servomotor a partir de la ecuación (6). Comprobar que se trata de un sistema de primer orden si se considera como salida la velocidad del eje de carga $\omega_l(t)$ del motor, y como entrada la tensión del circuito eléctrico con la que se alimenta el motor $v_m(t)$. (Nota: Considerar condiciones iniciales nulas)

(b) Identifica la dependencia de los distintos parámetros que definen la función de transferencia del sistema de primer orden (ganancia, constante de tiempo) con respecto a los parámetros que definen los elementos del motor. ¿De qué parámetros del modelo analítico (elementos del motor) dependerá la ganancia? ¿Y la constante de tiempo? (Ayuda: Utiliza el resultado de la cuestión anterior)

(c) Dados los valores numéricos de los parámetros que definen el motor, determina el valor de la ganancia y de la constante de tiempo del apartado anterior, para ello emplea los valores de la Tabla 1. Además, se sabe que $J_{eq} = \left(n_g {K_g}^2 J_m + J_l\right)$, $B_{eq} = \left(n_g {K_g}^2 B_m + B_l\right)$ y que $K_b = K_m$ y que $R_i = R_m$.

Symbol	Description	Value	Variation
V_{nom}	Motor nominal input voltage	6.0 V	
R_m	Motor armature resistance	2.6 Ω	± 12%
L_m	Motor armature inductance	0.18 mH	
k_t	Motor current-torque constant	7.68×10^{-3} N-m/A	± 12%
k_m	Motor back-emf constant	7.68×10^{-3} V/(rad/s)	± 12%
K_g	High-gear total gear ratio	70	
	Low-gear total gear ratio	14	
η_m	Motor efficiency	0.69	± 5%
η_g	Gearbox efficiency	0.90	± 10%
$J_{m,rotor}$	Rotor moment of inertia	3.90×10^{-7} kg-m^2	± 10%
J_{tach}	Tachometer moment of inertia	7.06×10^{-8} kg-m^2	± 10%
J_{eq}	High-gear equivalent moment of inertia without external load	2.087×10^{-3} kg-m^2	
	Low-gear equivalent moment of inertia without external load	9.7585×10^{-5} kg-m^2	
B_{eq}	High-gear Equivalent viscous damping coefficient	0.015 N-m/(rad/s)	
	Low-Gear Equivalent viscous damping coefficient	1.50×10^{-4} N-m/(rad/s)	
m_b	Mass of bar load	0.038 kg	
L_b	Length of bar load	0.1525 m	
m_d	Mass of disc load	0.04 kg	
r_d	Radius of disc load	0.05 m	
m_{max}	Maximum load mass	5 kg	
f_{max}	Maximum input voltage frequency	50 Hz	
I_{max}	Maximum input current	1 A	
ω_{max}	Maximum motor speed	628.3 rad/s	

Tabla 1. Especificaciones del servomotor

Symbol	Description	Value
K_{gi}	Internal gearbox ratio	14
$K_{ge,low}$	Internal gearbox ratio (low-gear)	1
$K_{ge,high}$	Internal gearbox ratio (high-gear)	5
m_{24}	Mass of 24-tooth gear	0.005 kg
m_{72}	Mass of 72-tooth gear	0.030 kg
m_{120}	Mass of 120-tooth gear	0.083 kg
r_{24}	Radius of 24-tooth gear	6.35×10^{-3} m
r_{72}	Radius of 72-tooth gear	0.019 m
r_{120}	Radius of 120-tooth gear	0.032 m

Symbol	Description	Value	Variation
K_{pot}	Potentiometer sensitivity	35.2 deg/V	± 2 %
K_{enc}	Encoder sensitivity	4096 counts/rev	
K_{tach}	Tachometer sensitivity	1.50 V/k_{RPM}	± 2 %

Tabla 1bis. Especificaciones del tren de engranaje y de los sensores de lectura

(d) Calcula la ganancia del sistema en estado estacionario K, así como su constante de tiempo τ, ante una entrada escalón de 3 unidades. Se sabe que la ganancia en estado estacionario se puede calcular como $K = \frac{y_{ss}-y_0}{x_{max}-x_{min}}$ donde $y_0 = y(0)$ e $y_{ss} = y(\infty)$ y donde x(t) es la señal de entrada.

(e) Calcula la respuesta temporal del sistema y(t) ante entrada escalón unitario, si al sistema cuya función de transferencia ha sido obtenida en (b), se le añade un retardo. Proporciona la respuesta analítica y dibújala con MATLAB.

(f) Calcula el diagrama de Bode del sistema a partir de la respuesta ante una entrada sinusoidal. Para poder calcular el diagrama de Bode, es necesario obtener la respuesta del sistema y(t) ante una entrada sinusoidal en la que se va variando la frecuencia de entrada de la señal sinusoidal y observando como varía la respuesta del sistema y(t) para cada caso. Si se conoce la función de transferencia del sistema, la magnitud del diagrama de Bode se puede calcular como $|G(j\omega)| = \left|\frac{\Omega_l(j\omega)}{V_m(j\omega)}\right| = \frac{K}{\sqrt{1+\tau^2\omega^2}}$ y la fase como $\angle G(j\omega) = -\text{atan}(\omega\tau)$ (Nota: Ver transparencia 17-Sesión 8 del tema 4).

Sesión 2: Identificación del sistema experimentalmente

En esta otra parte de la práctica, se identificará y obtendrá la función de transferencia del sistema de modo experimental, y posteriormente se realizará un breve análisis de su comportamiento que se comparará con el modelo analítico obtenido en la sesión anterior.

S2.1.-Configuración del motor

Para poder trabajar con el motor y poder analizar su respuesta conectándolo a un PC y empleando MATLAB y SIMULINK como herramientas de análisis y control, se proporcionan un conjunto de archivos y su descripción (ver Tabla 2), los cuales recogen la implementación del control en posición de un servomotor SRV02 del fabricante QUANSER.

Archivo	Descripción
setup_srv02_exp01_mdl.m	El *script* principal de MATLAB que establece los parámetros del motor y del sensor, así como su configuración de los parámetros del modelo. Es necesario ejecutar este archivo para poder realizar los experimentos de esta práctica.
config_srv02.m	Establece los parámetros de configuración del modelo del servomotor SRV02: *Rm*, *kt*, *km*, *Kg*, *eta_g*, *Beq*, *Jeq* y *eta_m;* y las constantes de calibración del sensor: K_POT, K_ENC y K_TACH, y las limitaciones del amplificador VMAX_AMP y IMAX_AMP.
calc_conversion_constants.m	Para calcular las conversiones de unidades de los parámetros calculados
q_srv02_mdl.mdl	Archivo de SIMULINK que implementa un control en bucle abierto para el sistema real SRV02 utilizando QUARC

Tabla 2. Archivos de configuración del servomotor

Los pasos para llevar a cabo la configuración del motor y poder llevar a cabo la experimentación se detallan a continuación:

1. Ejecuta MATLAB

2. Establece como carpeta actual de trabajo en MATLAB, el paquete de archivos proporcionados por el profesor para esta práctica (donde está el archivo **q_srv02_mdl.mdl**).

3. Abrir (doble clic) el archivo **q_srv02_mdl.mdl** de SIMULINK en el cual se muestra el modelo del del sistema *SRV02-ET* que representa el servomotor SRV02 que se emplea en la práctica.

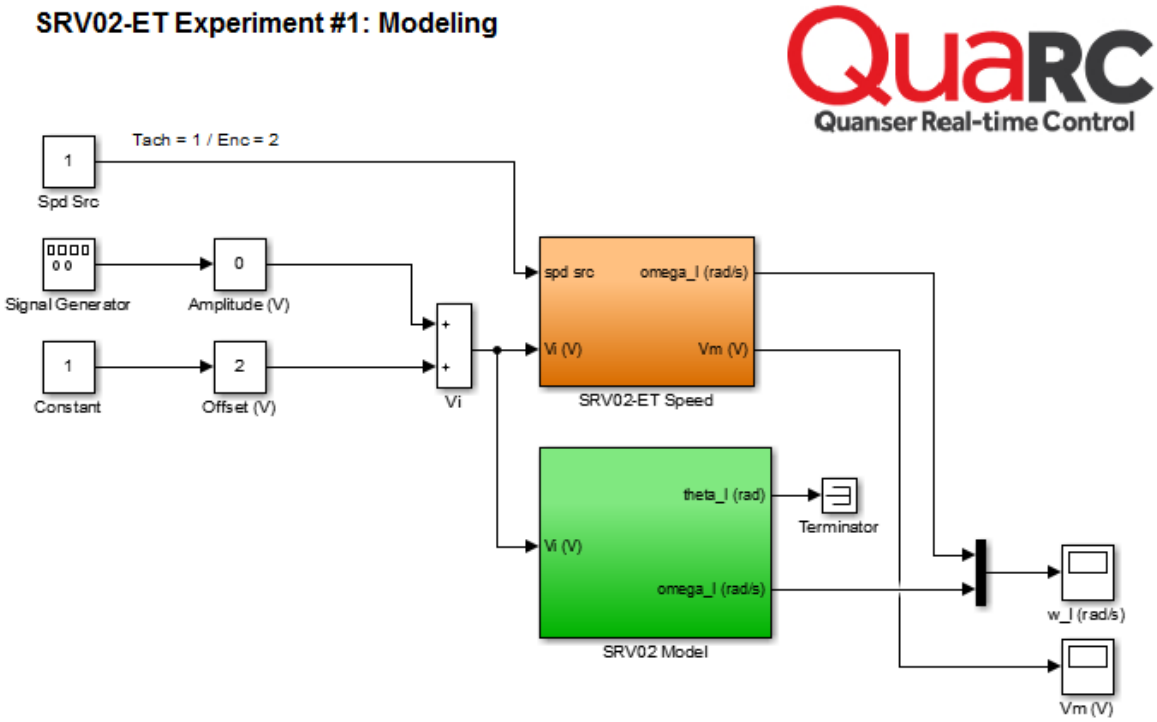


Figura 1. Modelo del servomotor en SIMULINK

4. Configurar el sensor para medir. La velocidad de salida en el eje del motor se puede medir usando varios sensores. En el bloque *Spd Src* del esquema **q_srv02_mdl,** se proporcionan las siguientes dos opciones: 1 para usar *tacómetro* y 2 para usar el *encoder*. Se recomienda utilizar la opción 2 para obtener mayor previsión en la medición de la velocidad.

5. Abrir (doble clic) el archivo **setup_srv02_exp01_mdl.m** para abrir la configuración del modelo del servomotor, mostrado en el esquema de SIMULINK **q_srv02_mdl**.

6. Una vez abierto, asegúrate de que el *script* esté configurado para coincidir con la configuración necesaria para el modelo de dispositivo real SRV02 (ya que otros modelos de QUANSER emplean otras configuraciones). La configuración deseada debe coincidir con la que se indica aquí debajo, y además el alumno se debe asegurar que la variable MODELING_TYPE está ajustada en 'MANUAL'.

```
%% SRV02 Configuration
% External Gear Configuration: set to 'HIGH' or 'LOW'
EXT_GEAR_CONFIG = 'HIGH';
% Encoder Type: set to 'E' or 'EHR'
ENCODER_TYPE = 'E';
% Is SRV02 equipped with Tachometer? (i.e. option T): set to 'YES' or 'NO'
TACH_OPTION = 'YES';
% Type of Load: set to 'NONE', 'DISC', or 'BAR'
LOAD_TYPE = 'DISC';
% Amplifier Gain: set VoltPAQ amplifier gain to 1
K_AMP = 1;
% Power Amplifier Type: set to 'VoltPAQ', 'UPM_1503', 'UPM_2405', or 'Q3'
AMP_TYPE = 'VoltPAQ';
% Digital-to-Analog Maximum Voltage (V)
VMAX_DAC = 10;
%
%% Lab Configuration
% Type of Controller: set it to 'AUTO', 'MANUAL'
MODELING_TYPE = 'MANUAL';
```

7. Ejecutar el *script* y entonces MATLAB deberá haber generado el mensaje que se muestra a continuación. (NOTA: Los parámetros que se muestran de ganancia y constante de tiempo, son los predeterminados inicialmente, por el fabricante y en ningún caso representan con precisión al sistema real).

```
Calculated SRV02 model parameter:
   K = 1 rad/s/V
   tau = 0.1 s
```

S2.2. Experimentación con el motor: Obtención de la respuesta temporal ante entrada escalón

El sistema *SRV02-ET* (Figura 1) contiene bloques tipo QUARC que interactúan con el motor real de CC y sus sensores. El sistema *SRV02 Model* utiliza un bloque de función de transferencia del SIMULINK para simular el sistema real. De este modo, tanto la velocidad del eje medida del motor real como la simulada pueden monitorizarse simultáneamente, dada una tensión de entrada.

Para ello, el alumno debe proceder siguiendo los siguientes pasos:

1. Ajustar el tipo de señal en el bloque *SRV02 Signal Generator* para generar una señal periódica.
   - Tipo de señal = cuadrada (*square*).
   - Amplitud = 1.
   - Frecuencia = 0,4 Hz.

2. Ajustar el bloque *Amplitude* a 1,5 V y *Offset* a 2.0 V.

3. Abrir y visualizar los gráficos (*plots*) de velocidad **w_1** (rad/s) y tensión de entrada del servomotor, **Vm** (V).

4. Hacer clic en QUARC | *Build Model* para compilar el esquema *Simulink*.

5. Ejecutar el diagrama (botón *Run*) durante 6 segundos de simulación. Los engranajes del servomotor deben girar en la misma dirección y alternando entre velocidades bajas y altas. Se deben generar gráficas similares a las mostradas en la Figura 2.

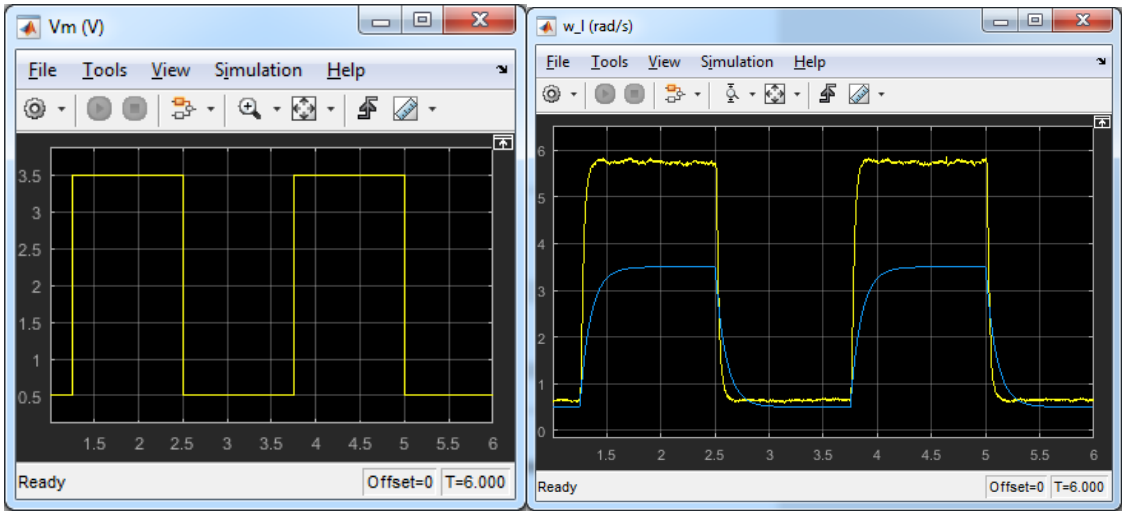


Figura 2. Entrada (izquierda) y Salida real y simulada en azul (derecha)

S2.3. Experimentación con el motor: Obtención de la ganancia en régimen permanente

Para encontrar la ganancia en estado estacionario del sistema, se requiere ejecutar el sistema con una entrada constante de voltaje. Para crear una tensión de entrada constante de 2V, el alumno debe proceder siguiendo los siguientes pasos:

1. Ajustar el tipo de señal en el bloque *SRV02 Signal Generator* para generar una señal periódica.
   - Tipo de señal = seno (*sine*).
   - Amplitud = 1.
   - Frecuencia = 0,0001 Hz.

2. Ajustar el bloque *Amplitude* a 0 V y *Offset* a 2.0 V.

3. Abrir y visualizar los gráficos (*plots*) de velocidad **w_l** (rad/s) y tensión de entrada del servomotor, **Vm** (V).

4. Hacer clic en QUARC | *Build Model* para compilar el esquema *Simulink*. Si se ha compilado correctamente, haz clic en el botón *Connect to Target*

5. Ejecutar el diagrama (botón *Run*) durante 5 segundos. Al ejecutar el diagrama se deben generar gráficas similares a las mostradas en la Figura 3.

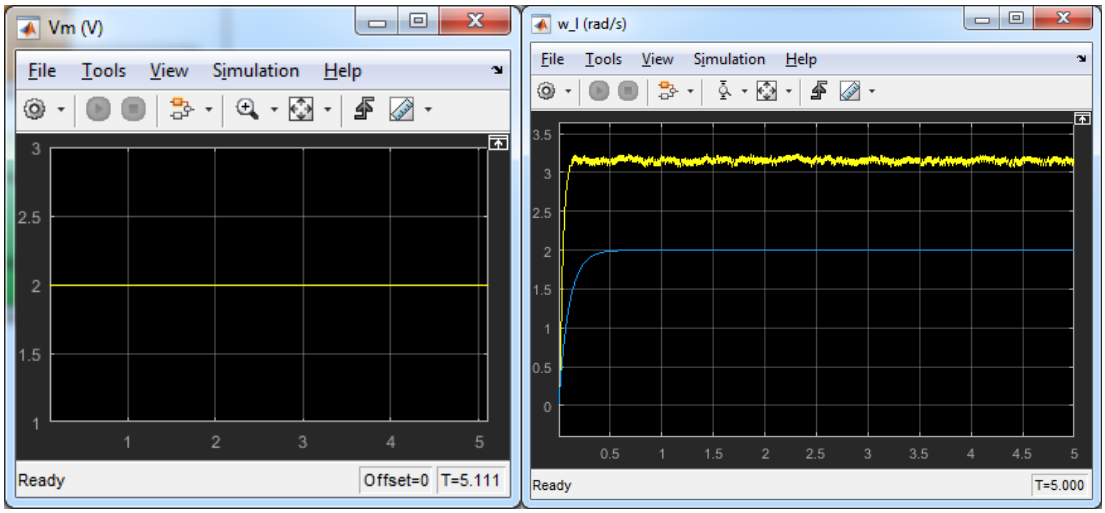


Figura 3. Entrada (izquierda) y Salida real y simulada en azul (derecha)

S2.4. Experimentación con el motor: Obtención de la respuesta frecuencial

La respuesta de un sistema en frecuencias ante una entrada sinusoidal es una señal sinusoidal con la misma frecuencia pero con distinta amplitud y ángulo de fase (ver transparencias 1-10 del tema 4 sesión 8). Para obtener la respuesta se procede manteniendo la amplitud de la señal de entrada constante y variando la frecuencia, como ya se comentó en teoría. La relación entre las amplitudes de salida y de entrada a una frecuencia determinada se puede utilizar para crear un diagrama de Bode. Entonces, la función de transferencia del sistema puede ser obtenida a partir de este diagrama de Bode (ver Figura 4).

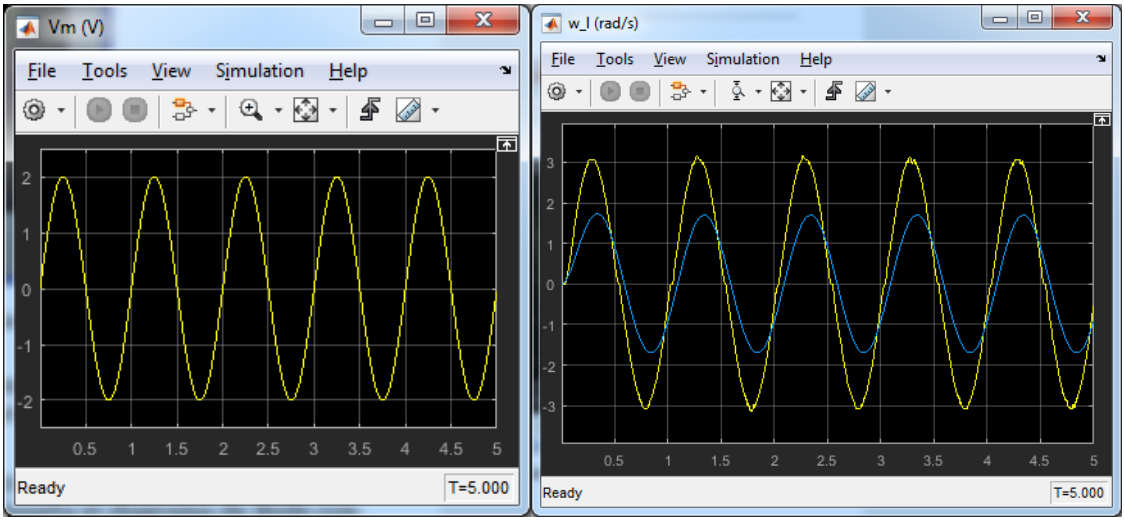


Figura 4. Entrada (izquierda) y Salida real y simulada en azul (derecha)

Ejercicio 2 (Sesión 2)

(a) Proceder como en la sección S2.2 e implementar un *script* (llamado ejercicio1) en MATLAB que genere un gráfico con la respuesta del sistema junto con la simulada (ver Figura 2). La velocidad de salida del motor se guardará en el *workspace* de Matlab en la variable **w_1**.

(b) En el mismo script, introduzca el código que calcule la ganancia en estado estacionario usando la respuesta ante entrada escalón medida e introdúcela en la Tabla 3 de resultados. Calcula también la constante de tiempo de la respuesta obtenida e introdúcela en la misma tabla. Compárala con los valores obtenidos en el ejercicio 1b de la sesión 1.

Sección	Descripción	Símbolo	Valor
Ejercicio 1b (Sesión 1)	Parámetros nominales del modelo Ganancia motor Constante de tiempo	 K τ	
Ejercicio 2a (Sesión 2)	Respuesta ante escalón Ganancia motor Constante de tiempo	 $K_{e,1}$ $\tau_{e,1}$	
Ejercicio 3d (Sesión 2)	Validación del modelo Ganancia motor Constante de tiempo	 $K_{e,2}$ $\tau_{e,2}$	
Ejercicio 4g (Sesión 2)	A partir de Bode (frecuencias) Ganancia motor Constante de tiempo	 $K_{e,3}$ $\tau_{e,3}$	

Tabla 3. Resultados en dominio del tiempo

Ejercicio 3 (Sesión 2)

(a) Modifique el paso 1 y 2 de la experimentación con el motor para ajustar la amplitud a 1V en vez 1,5V y el *Offset* a 1,5V en vez de a 2V. Y repita los pasos del 3 al 5 para obtener las gráficas de entrada y de salida que se implementaron en el ejercicio 1a (ejercicio anterior).

(b) Modifique el valor de ganancia K=1.25 (en lugar de K=1) y el valor de τ = 0,2 (en lugar del por defecto de de τ = 0,1) en el *workpsace* de MATLAB. Para que los cambios surtan efecto, acuérdese de actualizar el diagrama (opción *Simulation* → *Update Diagram*). Represente las gráficas de entrada, salida real y simulada y observe cómo cambia la simulación.

(c) Experimente variando los parámetros del modelo (ganancia y constante de tiempo) y en cada caso represente las gráficas de entrada, salida real y simulada, y compárelas entre sí. ¿Cómo afectan la ganancia y la constante de tiempo a la respuesta del sistema?

(d) Modifique y actualice el sistema a los valores de ganancia y constante de tiempo a los obtenidos en el ejercicio 1b de la sesión 1 y repita la simulación, volviendo a representar la entrada y salida. ¿La respuesta simulada obtenida se aproxima más o menos que la respuesta simulada en los casos anteriores experimentados en el ejercicio?

Ejercicio 4 (Sesión 2)

(a) Proceder como en la sección S2.4, considerando una amplitud de señal de entrada de 2V y variando la frecuencia y tomando mediciones de la velocidad del eje de carga, para introducirlas en la Tabla 4 de resultados. (Nota: La medición puede realizarse directamente desde el *plot*. Alternativamente, se puede

utilizar el *workspace* de MATLAB para encontrar la velocidad máxima utilizando la variable wl, donde wl (:, 1) es el vector de tiempo, wl (:, 2) es la velocidad medida, y wl (:, 3) es la velocidad simulada).

Frecuencia (Hz)	Amplitud (V)	Máxima velocidad (rad/s)	Ganancia |G(w)| rad/s/V	Ganancia |G(w)| rad/s/V, dB
0.0	**2.0**		(No es necesario)	
1.0	**2.0**			
2.0	**2.0**			
3.0	**2.0**			
4.0	**2.0**			
5.0	**2.0**			
6.0	**2.0**			
7.0	**2.0**			
8.0	**2.0**			

Tabla 4. Tabla de datos de diagrama de Bode

(b) Modifique el paso 1 y 2 de la experimentación mostrada en S.2.4 para ajustar la amplitud a 2V y el *Offset* a 0V. Y repita los pasos del 3 al 5 para obtener las gráficas de señal de entrada y señal de salida. Represente las gráficas de entrada, salida real y simulada.

(c) Mida la máxima velocidad positiva del eje para la frecuencia f = 1,0 Hz e introduzca el valor en la Tabla 4. Al igual que se ha explicado antes, esta medición se puede hacer directamente con la variable guardada *wl* en el *workspace* de MATLAB.

(d) Calcule la ganancia del sistema (en unidades lineales y dB) e introduce los resultados en la Tabla 4.

(e) Finalmente, modifique y actualice la frecuencia a f = 2.0 Hz ajustando el parámetro de frecuencia en el bloque *SRV02 Signal Generator*. Mida la velocidad máxima y calcule la ganancia, repitiendo este paso para cada uno de los valores de frecuencia de la Tabla 4.

(f) Construya un diagrama en magnitud de Bode a partir de las ganancias calculadas en dB y recogidas en la tabla 4 (Nota: Ignora la entrada f=0Hz). Dibuje el diagrama de Bode teórico a partir de los datos analíticos de la sección S1 (Ejercicio 1f) y compárelo con el obtenido aquí.

(g) Calcule la ganancia en estado estacionario K o ganancia máxima (Ver Sesión 8 del tema 4 de teoría), así como la constante de tiempo τ usando el gráfico de Bode obtenido en el apartado anterior a partir de la frecuencia de corte. Introduzca los datos en la última casilla de la Tabla 3, así como para el caso de la ganancia en la fila f=0Hz de la Tabla 4 (Nota: Recordad que la frecuencia de corte en un sistema con polos reales y sin ceros es la frecuencia en la que la ganancia es 3dB más pequeña que la ganancia máxima cuyo valor es $20\log|K|$).

(h) Observando el diagrama de Bode, estime los valores de margen de amplitud y margen de fase, y determine cómo es el sistema en cuanto a su estabilidad relativa (Ver Sesión 8 del tema 4, transparencias 37-40).

(Nota: utiliza el comando *ginput* para obtener valores de la gráfica en Matlab).