Programación en Matlab

Función de transferencia

Ing. Eddie Ángel
Sobrado Malpartida

-

Introducción

Para el ejemplo se va a partir de una descripción de la planta en forma de función de transferencia:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{0.2s^2 + 0.3s + 1}{(s^2 + 0.4s + 1)(s + 0.5)}$$

En Matlab, las funciones de transferencia se introducen dando el polinomio del numerador y polinomio del denominador:

```
>> num = [0.2 0.3 1];
>> den1 = [1 0.4 1];
>> den2 = [1 0.5];
```

El polinomio del denominador es el **producto** de dos términos. Para obtener el polinomio resultante se usa la convolucion (producto de polinomios).

```
>> den = conv(den1,den2)
```

Para ver los polos (o los ceros) de la función de transferencia, podemos usar:

```
>>roots(den)
>>roots(num)
```

Una forma más completa de convertir una función de transferencia dada por dos polinomios numerador y denominador, en un conjunto de factores de grado 1, correspondientes a los polos (z1, z2, z3) y ceros (c1, c2), de la forma:

$$H(s) = \frac{K(s+z1)(s+z2)}{(s+p1)(s+p2)(s+p3)}$$

es mediante el comando tf2zp:

```
>>[ceros,polos,gan] = tf2zp(num,den);
```

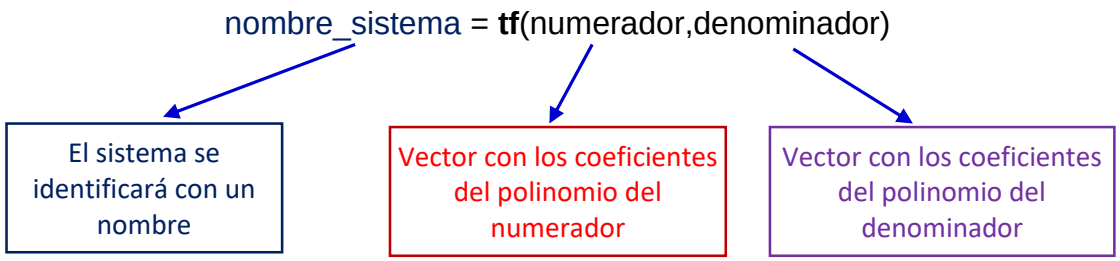
que devuelve un vector conteniendo los ceros de la función de transferencia, un vector conteniendo los polos, y un escalar correspondiente a la ganancia estática.
La función complementaria a esta también existe:

```
>>[N,D] = zp2tf (ceros,polos,gan);
```

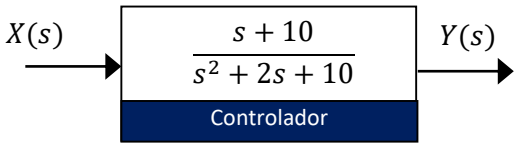
Cabe mencionar que el nombre de estas funciones es bastante descriptivo:
'tf-two-zp' procede de transfer-function to zero-pole form.

Representación en Función de transferencia

La forma de representar un sistema continuo es Matlab es mediante su función de transferencia en **s**, a través de la instrucción **tf.**



Ejemplo**:** Queremos representar el siguiente sistema continuo:



La instrucción de Matlab a utilizar será:

```
sis1= tf([1 10],[1 2 10])
```

La variable sis1 representará el sistema correspondiente a esa función de transferencia.

Respuesta a una entrada al sistema tipo escalón

La respuesta ante un escalón a la entrada se puede analizar en sistemas que tengan una descripción en forma de función de transferencia o una representación en el espacio de estados, generando un vector de tiempos y usando la función step.

Caben dos posibilidades:

a. Obtención de los valores numéricos de la respuesta:

En este caso lo que deseamos no es obtener el gráfico sino descargar en una variable el valor de la respuesta en cada instante de tiempo.

El formato para la instrucción es, en este caso:

```
sis1= tf([1 10],[1 2 10])
[y,t] = step(sis1);
```

y el resultado será el siguiente:

- El vector **t** contendrá los instantes de tiempo para los que se ha calculado el valor de la salida
- El vector **y** contendrá los valores de la salida correspondientes a cada instante de tiempo

Como comprobación, podemos consultar un valor cualquiera de la variable t y de la variable y en la ventana de comandos; por ejemplo, el componente 25 del vector *t* e *y:*

```
» t(25)
ans =
1.3252

» y(25)
ans =
1.1786
```

Vemos que el valor 25 corresponde al instante de tiempo 1.3252 segundos y que el valor de la señal de salida en ese instante de tiempo es 1.1786.
Disponer de los valores numéricos de los datos es útil para realizar cualquier tipo de operación matemática, como buscar el máximo, obtener el valor exacto en un instante de tiempo concreto, etc.

Otro ejemplo:

```
t = [0:.3:15]';
y = step(num,den,t);
plot (t,y);
title ('Respuesta a un escalon unitario');
xlabel ('tiempo(seg)');
grid;
```

b. Representación gráfica de la respuesta

No es necesario recuperar el resultado de la simulación ante escalón que realiza *step* para después representarlo con *plot*.
El propio comando *step*, si lo utilizamos sin parámetros de salida, realiza la representación gráfica. Es más, en este caso, la representación gráfica es algo más interactivo, puesto que, si hacemos 'click' con el botón izquierdo en algún punto de la gráfica, nos dice los valores correspondientes al tiempo y al valor de la salida en ese punto.

```
t = [0:.3:15]';
step(num,den,t);
title ('Respuesta a un escalon unitario');
xlabel ('tiempo(seg)');
grid;
```

c. Obtener la respuesta para un escalón no unitario

Lo visto anteriormente considera que al sistema se le aplica un escalón de valor uno. Para escalones no unitarios basta con multiplicar el sistema por el valor del escalón.
Por ejemplo, para obtener la respuesta a un escalón de 5 unidades bastará con teclear estas instrucciones:

```
step(5*sis1)
```

o bien:

```
[y,t] = step(5*sis1);
```

d. Obtener la respuesta para instantes de tiempo posteriores

En el ejemplo realizado, Matlab calcula la respuesta del sistema hasta el instante t = 6 segundos (se puede comprobar sobre el gráfico). Si se desea obtener la respuesta para instantes posteriores basta con especificar un valor para el tiempo final en la instrucción step.
Por ejemplo, si queremos obtener la respuesta ante escalón del sistema sis1 no hasta el instante t=6 sino hasta el instante t=12 deberíamos teclear el siguiente comando Matlab:

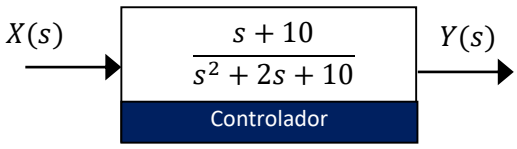
```
step(sis1, 12)
```

Respuesta a una entrada cualquiera

Al igual que la instrucción step nos ofrece la respuesta de un sistema a una señal escalón, también es posible obtener mediante Matlab la respuesta de un sistema ante una entrada cualquiera.
Para ello se utiliza la instrucción lsim que, al igual que la instrucción step, permite obtener los resultados de dos formas distintas:

a. Como una representación gráfica
b. Como valores numéricos

Hemos dicho que la instrucción lsim permite obtener la respuesta de un sistema ante una entrada cualquiera. El primer paso, antes de utilizar la instrucción, será definir la entrada a utilizar.
Por ejemplo, si deseamos conocer la respuesta del sistema sis1 definido anteriormente ante una entrada x(t) diferente al escalón, deberemos en primer lugar definir esa señal x(t):



Una señal cualquiera se definirá mediante dos vectores:
- Un vector de instantes de tiempo
- Un vector de valores para la señal en cada uno de esos instantes de tiempo

Con el objeto de representar la señal con la mayor precisión posible, es recomendable utilizar un gran número de valores o, lo que es lo mismo, hacer que los instantes de tiempo entre cada dos valores sean lo más pequeños posible.

Formatos para la instrucción *lsim*():

a. Si lo que se desea es la representación gráfica de la respuesta:

***lsim**(sistema, entrada, tiempo)*

sistema: El sistema se identificará con un nombre
entrada: Vector con los valores de la señal de entrada
tiempo : Vector con los instantes de tiempo de la señal de entrada

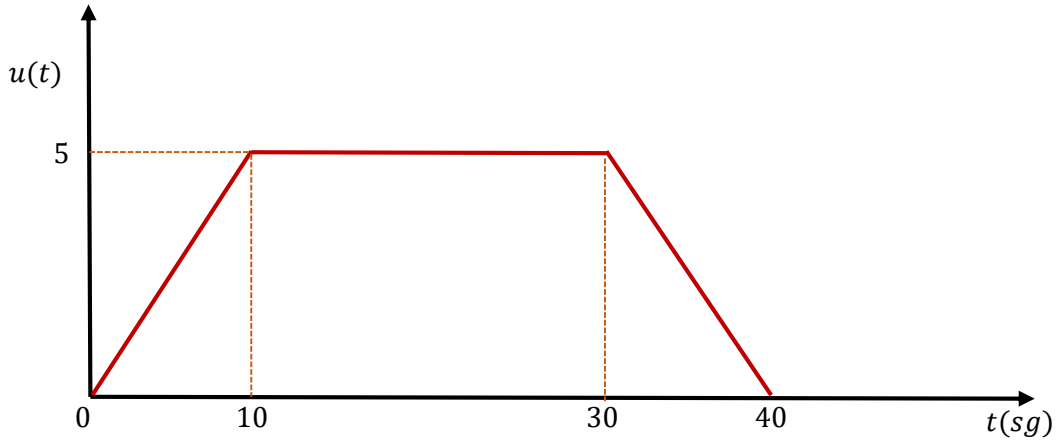
b. Si lo que se desea es obtener los valores numéricos de la respuesta:

***[y t]=lsim**(sistema, entrada, tiempo)*

y: vector de valores de la señal de salida
t: vector de instantes de la señal de salida

Al igual que con la instrucción **step**, los vectores *y* y *t* se pueden emplear para realizar cualquier operación matemática sobre esos datos.

Ejemplo**:** Obtendremos la respuesta del sistema **sis1** a la siguiente señal de entrada:



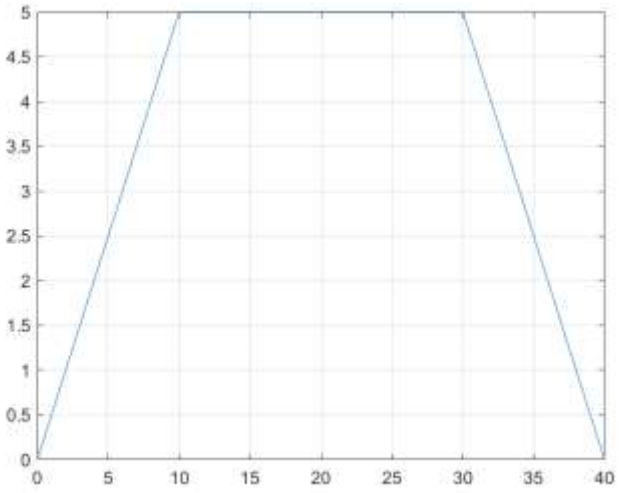
Como primer paso, debemos definir mediante dos vectores **t** (tiempo) y **u** (valores) la señal de entrada. De acuerdo con lo que se ha visto en prácticas anteriores, la definición de la señal se hará en tres tramos mediante las siguientes instrucciones de Matlab:

```
t1 = [0:0.1:10];        % de 0 a 10 seg. a intervalos de 0.1 seg.
u1 = 0.5*t1;            % primer tramo de la señal
t2 = [10.1:0.1:30];      % de 10.1 a 30 seg.
u2(1:200) = 5;           % segundo tramo de la señal
t3 = [30.1:0.1:40];      % de 30.1 a 40 seg.
u3 = 20 - 0.5*t3;       % tercer tramo de la señal
t = [t1, t2, t3];        % concatenación de los vectores de tiempo
u = [u1, u2, u3];        % concatenación de los vectores de datos
```

Es conveniente comprobar que se han definido correctamente las variables **u** y **t**. Para ello el procedimiento más inmediato es utilizar la orden **plot** de Matlab, con la que podemos representar la variable **u** (señal) en función de la variable **t** (tiempo):

```
» plot(t,u)
» grid
```

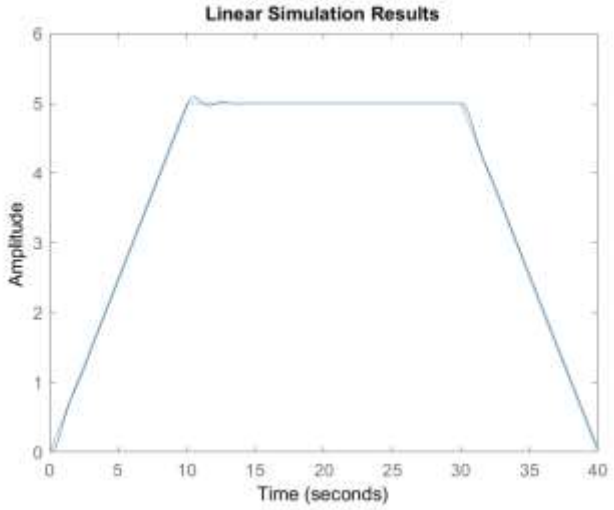
El resultado debe ser un gráfico similar al siguiente:



Una vez comprobado que los vectores se han definido correctamente, podemos lanzar la instrucción **lsim**:

```
» lsim(sis1,u,t)
```

Y el resultado será un gráfico con la respuesta que ofrece nuestro sistema **sis1** a la entrada anterior:



Podemos ver cómo la salida del sistema reproduce aproximadamente la entrada al mismo.