2º INGENIERÍA INDUSTRIAL

TEORÍA DE CIRCUITOS Y SISTEMAS

PRÁCTICA 7 SISTEMAS. UTILIDADES MATLAB

1. TRANSFORMADAS Y ANTITRANSFORMADAS

Matlab permite obtener transformadas y antitransformadas de Fourier, Laplace y Z mediante su módulo de matemática simbólica.

Procedimiento:

- Declarar una variable simbólica con la instrucción **syms**
- Obtener la transformada para una expresión definida utilizando la variable simbólica anterior

Instrucciones de Matlab correspondientes a cada una de las transformadas:

- **fourier** transformada de Fourier
- **ifourier** transformada inversa de Fourier
- **laplace** transformada de Laplace
- **ilaplace** transformada inversa de Laplace
- **ztrans** transformada Z
- **iztrans** transformada Z inversa

Ejemplo:

Obtendremos la antitransformada de Laplace de la siguiente expresión: $F(s) = \frac{2}{s \cdot (s + 0.5)}$

Las instrucciones de Matlab que utilizaremos serán:

```
» syms s
» ilaplace (2/(s*(s+0.5)))

ans =
4-4*exp(-1/2*t)
```

Por lo tanto, hemos obtenido como respuesta: $f(t) = 4 - 4 \cdot e^{-0.5t}$

Comprobación: haciendo el procedimiento inverso buscaremos la transformada de Laplace de f(t):

```
» syms t
» laplace (4-4*exp(-0.5*t))

ans =
 4/s-4/(s+1/2)
```

Se obtiene como resultado:

$$F(s) = \frac{4}{s} - \frac{4}{(s + 0.5)} = \frac{4(s + 0.5)}{s \cdot (s + 0.5)} - \frac{4 \cdot s}{s \cdot (s + 0.5)} = \frac{2}{s \cdot (s + 0.5)}$$

... que es la misma función F(s) de partida

Como ejercicio, se obtendrán antitransformadas de Laplace para las siguientes funciones:

$$F_1(s) = \frac{s^2 + 2s + 3}{(s+1)^3} \qquad f_1(t) = t^2 \cdot e^{-t} + e^{-t}$$

$$F_2(s) = \frac{5s + 5}{s \cdot (s^2 + 2s + 5)} \qquad f_2(t) = 1 - e^{-t} \cdot [\cos(2t) - 2 \cdot sen(2t)]$$

Y la antitransformada z de la siguiente función:

$$X(z) = \frac{z}{(z-1)^2 \cdot (z-2)} \qquad x_k = 2^k - 1 - k$$

2. REPRESENTACIÓN DE SISTEMAS CONTINUOS

Al igual que Simulink, Matlab también permite obtener la respuesta de sistemas continuos y discretos ante distintas señales de entrada.

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

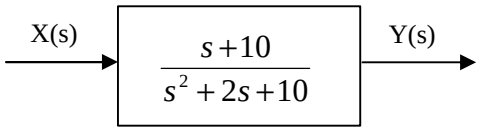
Forma de utilizar la instrucción tf:

```
nombre_sistema = tf (numerador, denominador)
```

cada sistema que creemos se identificará mediante su nombre	vector con los coeficientes del polinomio del numerador	vector con los coeficientes del polinomio del denominador

Ejemplo:

Queremos representar el siguiente sistema continuo:



La instrucción de Matlab a utilizar será:

```
» sis1 = tf([1 10], [1 2 10])

Transfer function:
     s + 10
---------------
s^2 + 2 s + 10
```

A partir de este momento, la variable sis1 representará en Matlab el sistema correspondiente a esa función de transferencia.

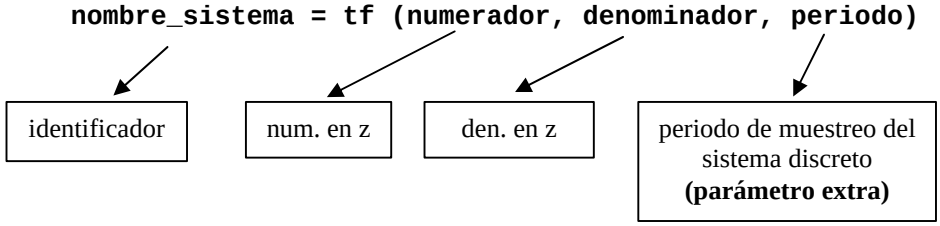
3. REPRESENTACIÓN DE SISTEMAS DISCRETOS

Un sistema discreto se representa en Matlab mediante su función de transferencia en **z**. Caben dos posibilidades:

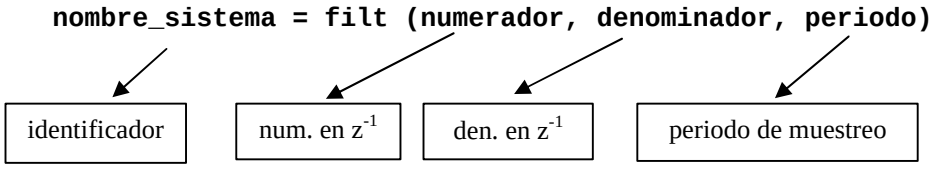
- Instrucción **tf**: permite escribir la función de transferencia en potencias positivas de z
- Instrucción **filt**: permite escribir la función de transferencia en potencias negativas de z

El formato con el que se deben especificar los datos es el siguiente:

- **Instrucción tf:** se utiliza el mismo formato que para sistemas continuos pero se añade un parámetro más: el periodo de muestreo:

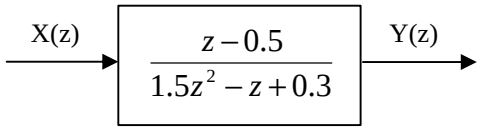


- **Instrucción filt:** también utiliza el mismo formato pero los coeficientes de numerador y denominador están expresados en potencias negativas



Ejemplo:

Deseamos representar el siguiente sistema discreto:



En el que supondremos que el periodo de muestreo es de 0,1 segundos.

- Empleando la instrucción **tf**, deberíamos escribir el siguiente código Matlab:

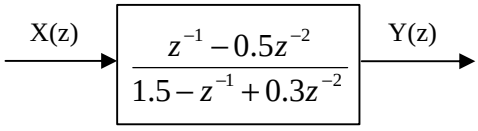
```
» sis2 = tf([1 -.5], [1.5 -1 .3], .1)

Transfer function:
     z - 0.5
-----------------
1.5 z^2 - z + 0.3

Sampling time: 0.1
```

- Empleando la instrucción **filt**, deberíamos en primer lugar expresar la función de transferencia en potencias negativas de z multimplicando numerador y denominador por z^{-2}:

$$\frac{z-0.5}{1.5z^2-z+0.3}=\frac{z^{-1}-0.5z^{-2}}{1.5-z^{-1}+0.3z^{-2}}$$



Una vez hecho esto, la orden Matlab a teclear es la siguiente:

```
» sis3 = filt([0 1 -.5], [1.5 -1 .3], .1)

Transfer function:
   z^-1 - 0.5 z^-2
--------------------
1.5 - z^-1 + 0.3 z^-2

Sampling time: 0.1
```

Las funciones **sis2** y **sis3** recién creadas se comportarán, por tanto, exactamente igual.

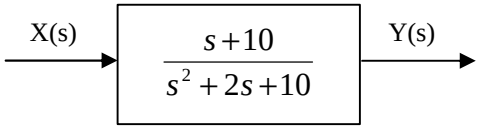
4. RESPUESTA A LA SEÑAL ESCALÓN

La instrucción **step** sirve para calcular la respuesta a escalón de cualquier sistema previamente definido. Caben dos posibilidades:

- Obtener la representación gráfica de la respuesta
- Obtener los valores numéricos de la respuesta

Representación gráfica de la respuesta

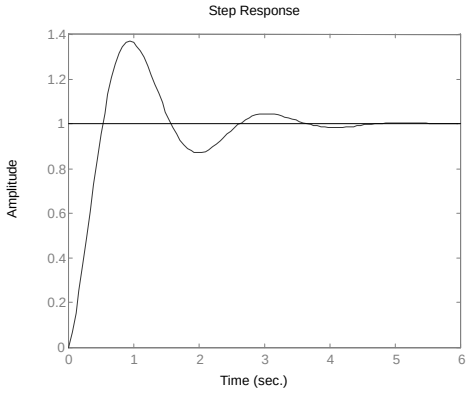
Obtendremos como ejemplo la respuesta a escalón del sistema continuo definido anteriormente:



Si suponemos que cuando creamos el sistema con la instrucción **tf** le dimos el nombre **sis1**, entonces la instrucción Matlab a teclear será la siguiente:

```
» step(sis1)
```

Y el resultado será el siguiente gráfico:



El gráfico representa el valor de la salida del sistema ante entrada escalón

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:

```
» [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

NOTA: recordemos que el punto y coma al final de la expresión sirve para que Matlab no muestre en pantalla el resultado de la operación; en otro caso habrían aparecido las ristras de valores de las variables.

Como comprobación, podemos consultar un valor cualquiera de la variable t y de la variable y; por ejemplo el valor que hace el número 25 de todos los calculados:

```
» 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. Podemos comprobar estos valores aproximadamente sobre el gráfico de la respuesta que obtuvimos antes.

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.

Forma de 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);
```

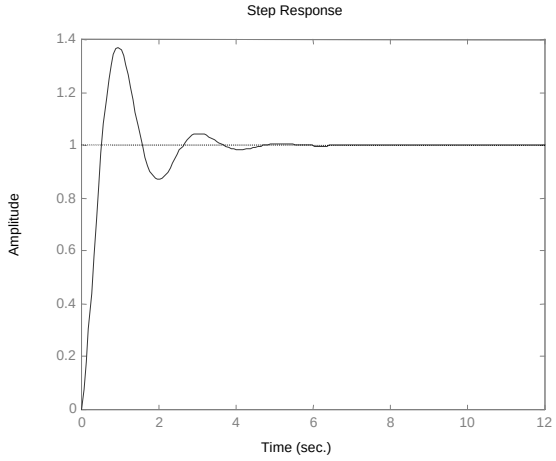
Forma de obtener la respuesta para instantes de tiempo posteriores

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 qeremos 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)
```

Y el resultado sería el que mostramos en el gráfico que aparece a continuación:



NOTA: Todo lo visto para sistemas continuos es directamente aplicable para sistemas discretos. Se recomienda probar la instrucción **step** con los sistemas **sis2** y **sis3** creados anteriormente.

5. RESPUESTA A UNA SEÑAL CUALQUIERA

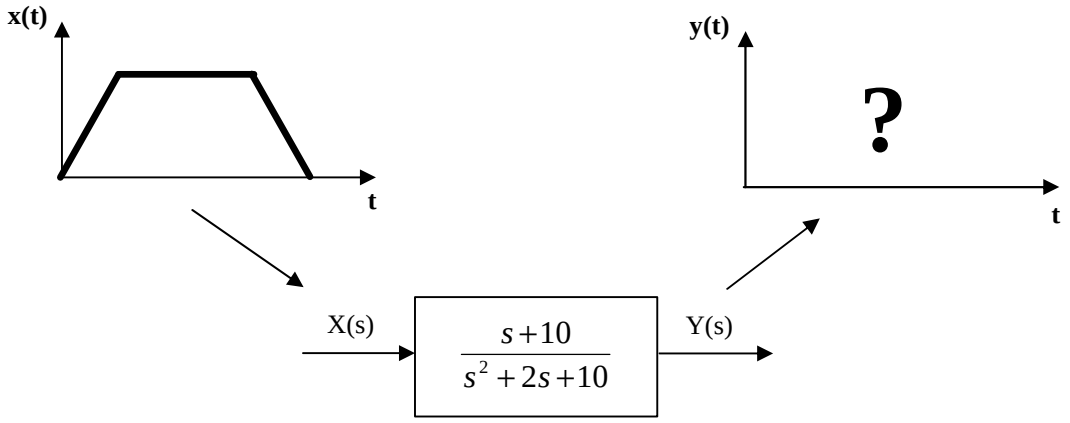
Al igual que la instrucción step nos ofrece la respuesta de un sistema a una señal escalón, también es posible obtener mediante Matlab la respuesta de un sistema ante una entrada cualquiera.

Para ello se utiliza la instrucción **lsim** que, al igual que la instrucción step, permite obtener los resultados de dos formas distintas:

- Como una representación gráfica
- 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) como la representada en la figura, deberemos en primer lugar definir esa señal x(t):



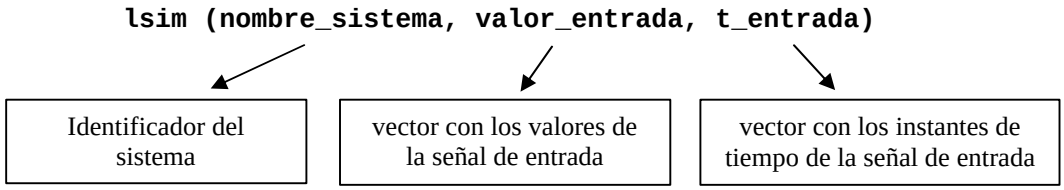
Una señal cualquiera se definirá mediante dos vectores:

- Un vector de instantes de tiempo
- Un vector de valores para la señal en cada uno de esos instantes de tiempo

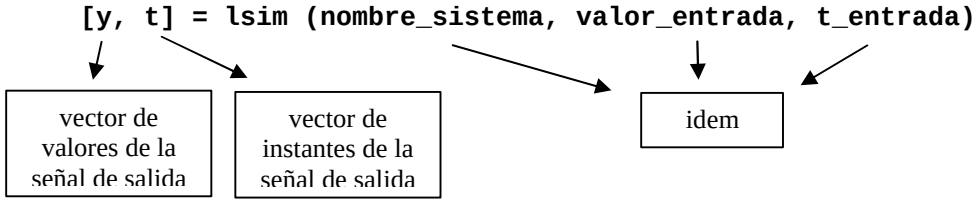
Con el objeto de representar la señal con la mayor precisión posible, es recomendable utilizar un gran número de valores o, lo que es lo mismo, hacer que los instantes de tiempo entre cada dos valores sean lo más pequeños posible.

Formatos para la instrucción lsim:

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



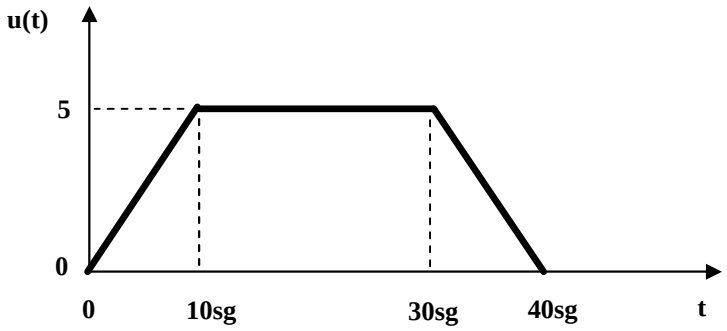
- Si lo que se desea es obtener los valores numéricos de la respuesta:



Al igual que con la instrucción step, los vectores y y t definen la señal de salida; de este modo es posible realizar cualquier operación matemática sobre esos datos.

Ejemplo:

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



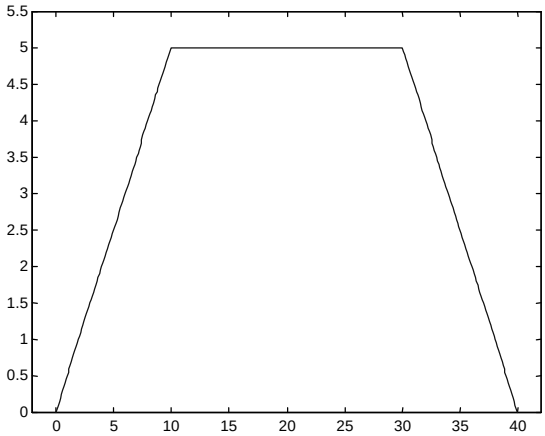
Como primer paso, debemos definir mediante dos vectores **t** (tiempo) y **u** (valores) la señal de entrada. De acuerdo con lo que se ha visto en prácticas anteriores, la definición de la señal se hará en tres tramos mediante las siguientes instrucciones de Matlab:

```
» 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)
```

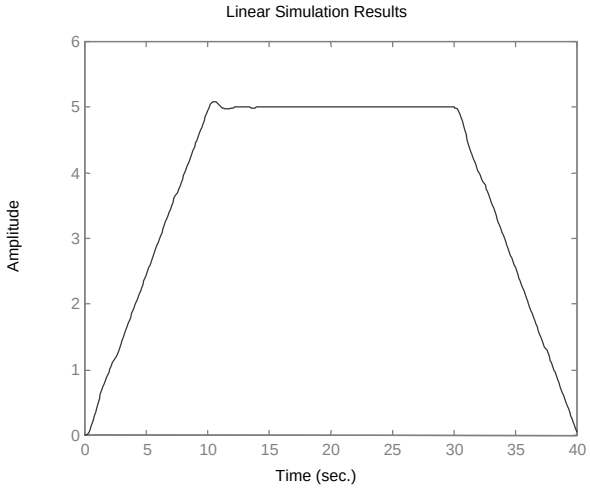
El resultado debe ser un gráfico similar al siguiente:



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:

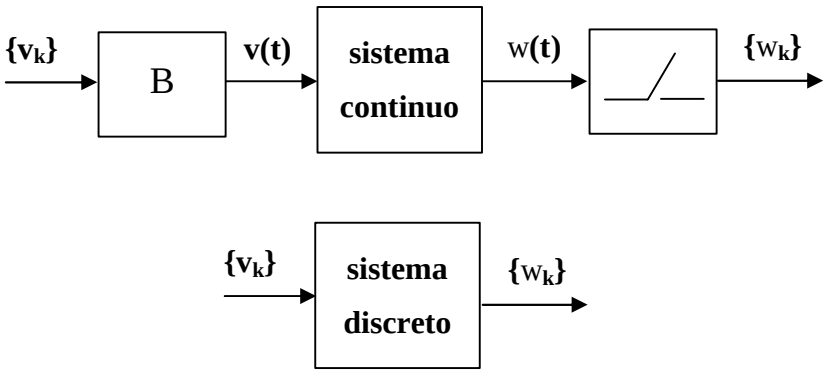


Podemos ver cómo la salida del sistema reproduce aproximadamente la entrada al mismo.

NOTA: Lo visto para sistemas continuos es directamente aplicable para sistemas discretos pero con una matización: las variables **t** y **u** se deben definir a intervalos de tiempo iguales al periodo de muestreo. Se recomienda probar la instrucción **lsim** con los sistemas **sis2** y **sis3** creados anteriormente.

6. DISCRETIZACIÓN DE SISTEMAS CONTINUOS

Matlab también dispone de herramientas para la obtención del sistema discreto equivalente a un sistema continuo dado:



Este proceso lo sabemos resolver manualmente mediante la fórmula de los residuos, y los resultados dependen del tipo de bloqueador empleado: de orden cero, de orden uno, etc.

La instrucción de Matlab que calcula el equivalente discreto para un sistema continuo dado es **c2d**.

El formato más simple para la instrucción **c2d** es el siguiente:

```
sist_discreto = c2d (sist_continuo, t_muestreo)
```

nombre del sistema discreto a crear	nombre del sistema continuo de partida	tiempo o periodo de muestreo

En este formato, la instrucción c2d asume que el bloqueador empleado es un bloqueador de orden cero.

Ejemplo:

Se desea obtener el equivalente discreto para el sistema continuo sis1 con el que venimos trabajando considerando un bloqueador de orden cero y un periodo de muestreo de 0.2 segundos. La instrucción Matlab a teclear sería:

```
» sis1d = c2d(sis1, .2)

Transfer function:
 0.3243 z - 0.005408
----------------------
z^2 - 1.351 z + 0.6703

Sampling time: 0.2
```

Podemos ver como Matlab nos devuelve la función de transferencia correspondiente al sistema discreto equivalente, que en este caso hemos llamado **sis1d**.

Utilización de un bloqueador de orden uno:

En el caso de utilizar un bloqueador de orden uno, el formato de la instrucción **c2d** cambia y es necesario añadir un parámetro más que especifica el tipo de bloqueador a utilizar. En el caso de nuestro sistema **sis1**, la expresión Matlab sería:

```
» sis1d = c2d(sis1, .2, 'foh')

Transfer function:
0.1444 z^2 + 0.2003 z - 0.02582
-------------------------------
    z^2 - 1.351 z + 0.6703

Sampling time: 0.2
```

Donde **'foh'** significa *first order hold* o, lo que es lo mismo, bloqueador de orden uno.

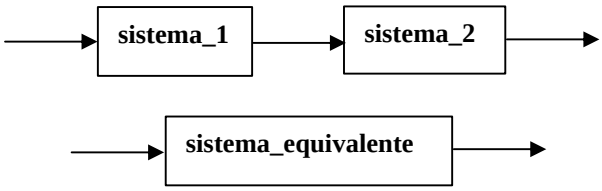
Podemos ver como la función de transferencia discreta obtenida es bastante distinta.

7. REDUCCIÓN DE DIAGRAMAS DE BLOQUES CON MATLAB

Matlab también permite obtener sistemas equivalentes para las principales combinaciones de bloques: en serie, en paralelo y en realimentación. De este modo, es posible reducir los diagramas de bloques.

Equivalente de dos bloques en serie:

Es el producto de los dos bloques, en Matlab se obtiene con el comando **series**:

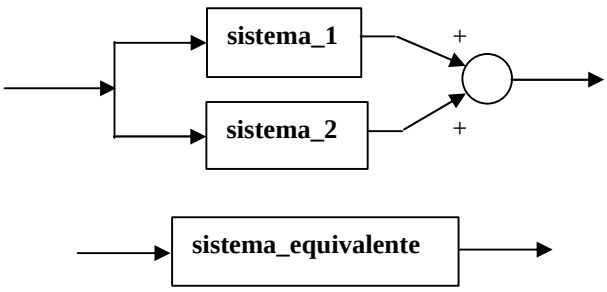


Supuestos definidos los sistemas sistema_1 y sistema_2, teclearíamos en Matlab:

```
» sistema_equivalente = series (sistema_1, sistema_2)
```

Equivalente de dos bloques en paralelo:

Es la suma de los dos bloques, en Matlab se obtiene con el comando **parallel**:

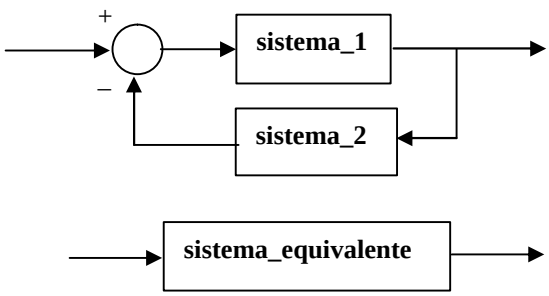


Supuestos definidos los sistemas sistema_1 y sistema_2, teclearíamos en Matlab:

```
» sistema_equivalente = parallel (sistema_1, sistema_2)
```

Equivalente para un esquema de realimentación:

Se calcula en Matlab mediante el comando **feedback**:

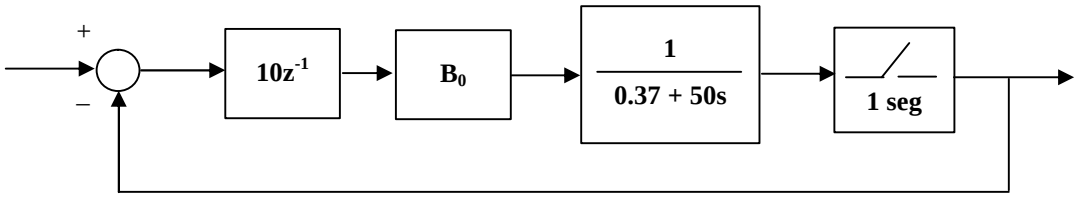


Supuestos definidos los sistemas sistema_1 y sistema_2, teclearíamos en Matlab:

```
» sistema_equivalente = feedback (sistema_1, sistema_2)
```

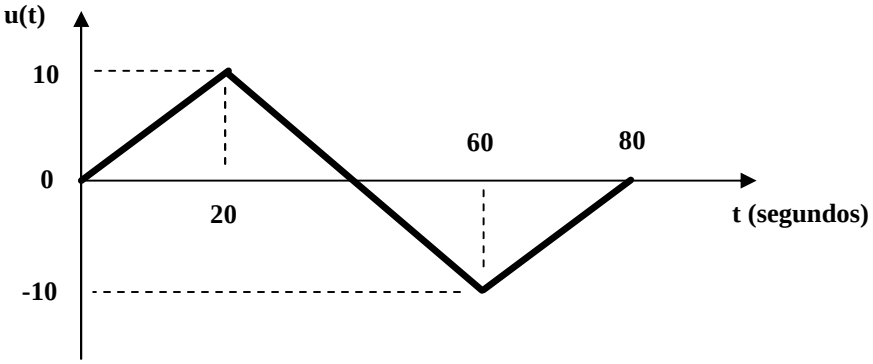
EJERCICIO: RESOLUCIÓN DE UN PROBLEMA COMPLETO CON MATLAB

Dado el siguiente sistema:



Se pide:

- Obtener el equivalente discreto para la parte continua
- Reducir el diagrama hasta obtener la función de transferencia total que relaciona entrada y salida
- Representar gráficamente la respuesta del sistema a un escalón de 10 unidades
- Representar la respuesta del sistema ante la siguiente señal de entrada:



NOTA: en este caso las variables **t** y **u** se deben definir con un intervalo de 1 segundo, por ser ese el periodo de muestreo del sistema discreto resultante.

A obtener con Matlab:

- Gráfico de la señal de respuesta ante escalón de 10 unidades.
- Gráfico de la señal de respuesta ante la señal de entrada del ejemplo.
  (Ambos gráficos deben incluir el nombre del alumno como título)