Práctica 3

5/04/2024
Fundamentos de Control

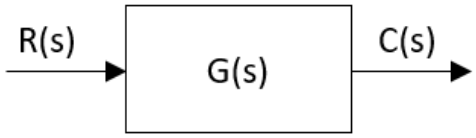
Juan López Puebla

Diagramas de bloques

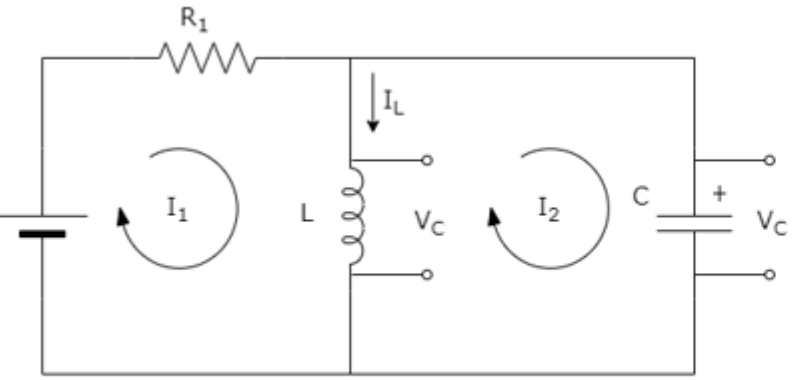
De conexiones entre modelos y reducción de bloques

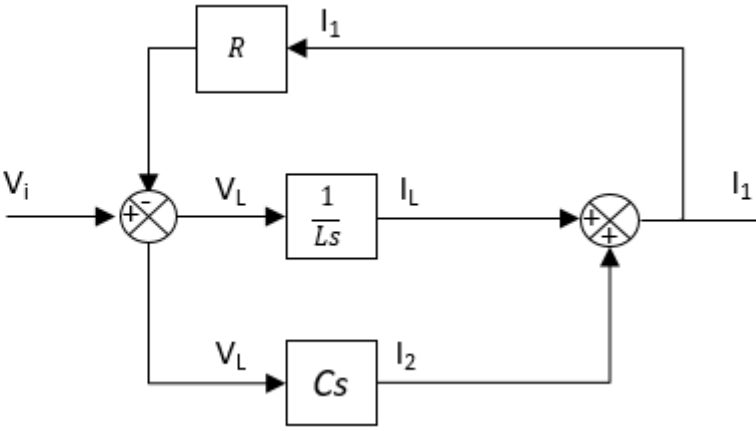
Table of Contents

Un sistema en general, y más concretamente un sistema de control, puede constar de varios componentes o elementos. Para representar la función que realiza cada uno de los componentes y sus relaciones (flujo de señales) se emplea un **diagrama de bloques**. En un diagrama de bloques, cada **bloque** representa la operación matemática (expresada como **función de transferencia**) que se aplica a la señal de entrada para producir su salida, y se emplean **flechas** para indicar la dirección del flujo de las señales. La siguiente imagen muestra un ejemplo de estos elementos:



Hay que tener en cuenta que $C(s) = G(s)R(s)$, o como ya sabemos, $G(s) = \frac{C(s)}{R(s)}$, siendo $G(s)$ la función de transferencia representada por el bloque en cuestión. La siguiente imagen muestra un diagrama de bloques un poco más elaborado. En concreto, a la izquierda se presenta un circuito RLC, y a la derecha el diagrama de bloques que lo modela con entrada $V_i(s)$ y salida $I_1(s)$:





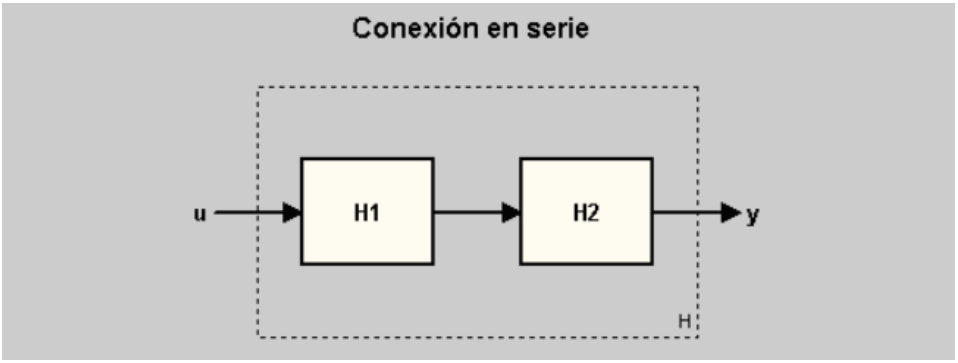
Si se conoce la relación matemática y funcional de cada uno de los componentes del sistema, su diagrama de bloques equivalente puede **reducirse** empleando el **algebra de bloques**, obteniendose un único bloque a analizar.

Dada su simplicidad y versatilidad, los diagramas de bloques resultan una herramienta comúnmente empleada por los ingenieros de control para modelar todo tipo de sistemas. MATLAB, como no podía ser de otra manera, nos facilita una serie de comandos para construir y reducir diagramas de bloques. ¡Vamos allá!

1. Conexión entre modelos

1.1 Conexión en serie o cascada

La **conexión en serie** o **cascada** entre modelos equivale al producto de las correspondientes funciones de transferencia:



Dados dos modelos, *H1* y *H2*, de dos sistemas conectados en serie, su modelo equivalente, *H*, podría ser obtenido en la forma siguiente:

```
H = series(H1, H2);
```

O de manera equivalente empleando operaciones aritméticas:

```
H = H1 * H2;
```

Tarea 1: Definir los siguientes modelos $H_1(s)$ y $H_2(s)$ y realizar su conexión en serie usando ambos métodos, mediante el comando **`series`** y mediante la multiplicación, comprobando que se obtiene el mismo resultado:

$$H_1(s) = \frac{2s+1}{s^2+5s+3} \qquad H_2(s) = \frac{1}{s+5}$$

```
% Tu código aquí
H1 = tf([2 1],[1 5 3]);
H2 = tf([1],[1 5]);
H3 = series(H1,H2)
```

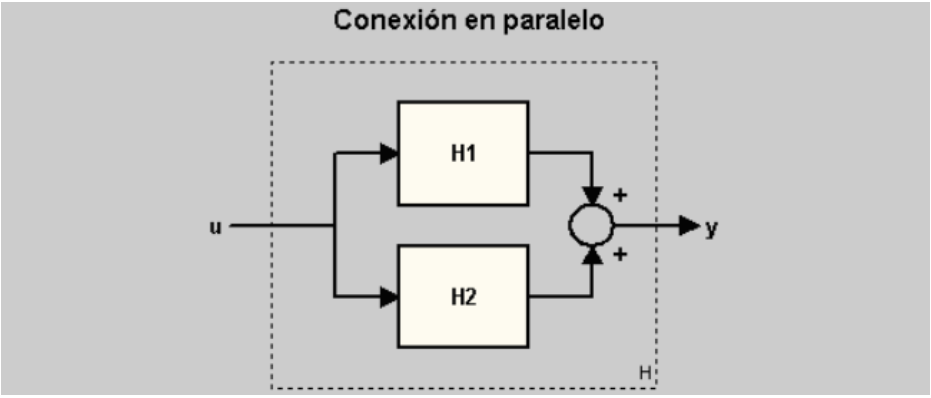
```
H3 =

          2 s + 1
  ------------------------
  s^3 + 10 s^2 + 28 s + 15

Continuous-time transfer function.
Model Properties
```

Resultado esperado:

```
H3 =

          2 s + 1
  ------------------------
  s^3 + 10 s^2 + 28 s + 15

Continuous-time transfer function.
```

1.2 Conexión en paralelo

La **conexión en paralelo** entre modelos equivale a la suma de las correspondientes funciones de transferencia.



Dados dos modelos, *H1* y *H2*, de dos sistemas conectados en paralelo, su modelo equivalente, *H*, podría ser obtenido en la forma siguiente:

```
H = parallel(H1, H2);
```

O de manera equivalente:

```
H = H1 + H2;
```

Tarea 2: Definir los siguientes modelos y realizar su conexión en paralelo usando tanto **`parallel`** como aritméticamente, comprobando que se obtiene el mismo resultado:

$$H_1(s) = \frac{2s+1}{s^2+5s+3} \qquad H_2(s) = \frac{1}{s+5}$$

```
% Tu código aquí
H1 = tf([2 1],[1 5 3]);
H2 = tf([1],[1 5]);
H3 = parallel(H1,H2)
```

```
H3 =

      3 s^2 + 16 s + 8
  ------------------------
  s^3 + 10 s^2 + 28 s + 15

Continuous-time transfer function.
Model Properties
```

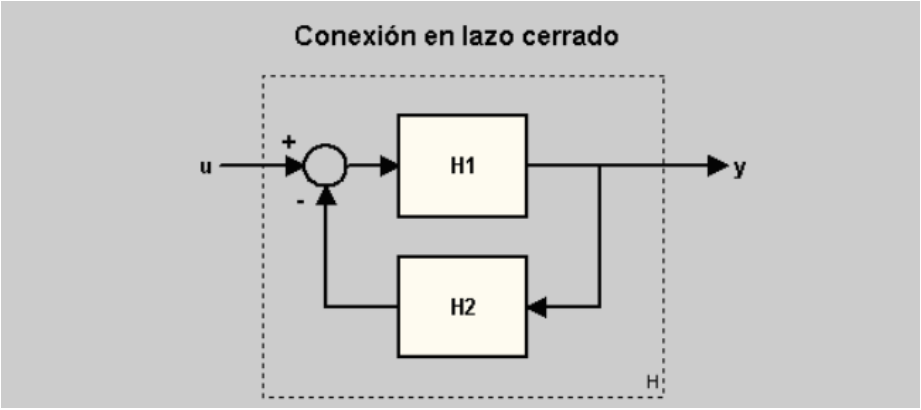
```

```

Resultado esperado:

```
H3 =

      3 s^2 + 16 s + 8
  ------------------------
  s^3 + 10 s^2 + 28 s + 15

Continuous-time transfer function.
```

1.3 Conexión entre modelos con realimentación

En este apartado se contempla la clásica conexión de dos modelos constituyendo un sistema realimentado.



Dados dos modelos, *H1* y *H2*, que constituyen un sistema realimentado con **realimentación negativa**, su modelo equivalente, *H*, podría ser obtenido en la forma siguiente:

```
H = feedback(H1, H2);
```

Este procedimiento es equivalente a:

```
H = feedback(H1, H2, -1);
```

De igual modo, empleando operaciones aritméticas:

```
H = H1/(1+H1*H2);
```

Dados dos modelos, *H1* y *H2*, que constituyen un sistema realimentado con **realimentación positiva**, su modelo equivalente, *H*, podría ser obtenido en la forma siguiente:

```
H = feedback(H1, H2, +1);
```

O de manera equivalente:

```
H = H1/(1-H1*H2);
```

Cabe señalar que, tanto en el uso de las funciones **`series`**, **`parallel`** y **`feedback`** como en los cálculos aritméticos equivalentes a ellas que se acaban de describir, si uno de los modelos intervinientes tiene formato *`tf`* o *`zpk`* y el otro modelo es representable por un valor constante, dicho valor se podría usar directamente como parámetro de entrada de la correspondiente función o en el cálculo aritmético en cuestión, sin necesidad de dar también a este modelo un formato *`tf`* o *`zpk`*.

Por ejemplo si tenemos dos modelos $H_1(s) = 3$ y $H_2(s) = \frac{2}{s+3}$ en serie:

```
H1 = 3;
H2 = tf(2,[1 3]);
H = series(H1,H2);
% o bien
H = series(3,H2);
% o bien
H = 3*H2
```

Tarea 3: Definir los siguientes modelos y realizar su conexión con realimentación negativa usando tanto **`feedback`** como aritméticamente, comprobando que se obtiene el mismo resultado:

$$\mathrm{H1} = \frac{s+5}{(s+3)(s+2)} \qquad\qquad \mathrm{H2} = \frac{3(s+1)}{s+5}$$

*Pista: Cuando se obtienen funciones de transferencia fruto de un comando distinto de **`zpk`** o **`tf`**, o usando aritmética de bloques, puede que la función de transferencia resultante tenga factores comunes en su numerador y denominador (esto es, ceros y polos con el mismo valor). Para simplificarlos podéis hacer uso del comando **`minreal`** en la forma siguiente: `Sysn = minreal(Sys)`. Usa el comando **`help`** para más info.*

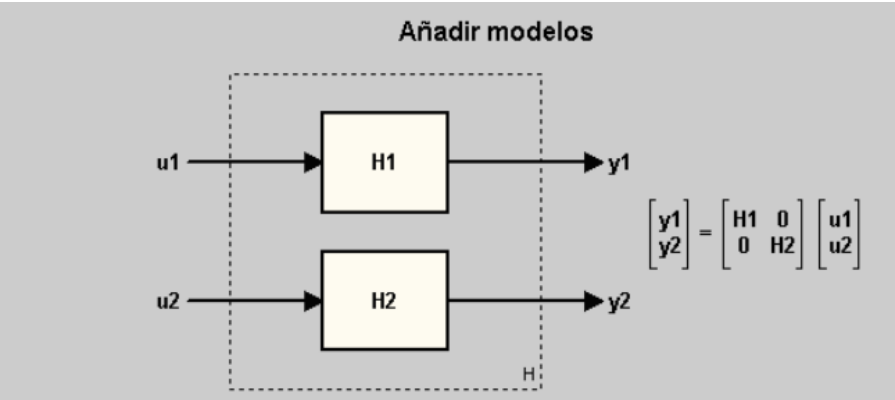
```
% Tu código aquí
s = tf('s');
H1 = (s+5)/((s+3)*(s+2));
H2 = (3*(s+1))/(s+5);
HF = feedback(H1,H2);
H = zpk(HF);
H =zpk(H1/(1+H1*H2));
minreal(H)
```

```
ans =

          (s+5)^2
  -------------------------
  (s+6.646) (s+5) (s+1.354)

Continuous-time zero/pole/gain model.
Model Properties
```

```
%Como podemos observar los resulados son los mismos
```

2. Agrupación de modelos

MATLAB permite agrupar modelos de sistemas SISO mediante la construcción de una matriz diagonal de agregación donde las filas son salidas y las columnas son entradas, resultando en un sistema de múltiples entradas y múltiples salidas (MIMO).



Para agrupar los modelos de dos sistemas SISO, *H1* y *H2*, para constituir el modelo *H* de un sistema con dos entradas y dos salidas en la forma indicada en la figura anterior, se podría actuar en la forma siguiente:

```
H = append(H1, H2);
```

Como resultado de esta actuación, el valor asignado a H es la matriz de transferencia del correspondiente sistema MIMO:

$$H = \begin{bmatrix} H1 & 0 \\ 0 & H2 \end{bmatrix}$$

Un procedimiento similar sería aplicable para obtener modelos de sistemas MIMO generados como agrupación de tres o más sistemas SISO, para lo que habría que añadirlos como parámetros de entrada adicionales al comando **append**.

Es de reseñar que, en el uso del comando **append**, si al menos uno de los modelos intervinientes en la agrupación de sistemas SISO considerada tiene formato *tf* o *zpk* y uno o varios de los restantes modelos involucrados en dicha agrupación es/son representable/s por un valor constante, dicho valor se podría usar directamente como parámetro de entrada de la correspondiente función, sin necesidad de dar también a este/estos modelo/s un formato *tf* o *zpk*.

3. Reducción de diagramas de bloques

MATLAB provee un poderoso comando para reducir un diagrama de bloques arbitrariamente complejo: **`connect`**. Para aplicar este comando hay que realizar una identificación (numeración creciente) de los bloques del diagrama, para lo que hay que tener en cuenta las siguientes consideraciones:

- Será imprescindible que el diagrama considerado posea, para cada una de las variables de entrada del sistema global, un bloque que tenga a ésta como entrada y, para cada una de las variables de salida del sistema global, un bloque que tenga a ésta como salida.
- En caso de ser necesario para cumplir estas condiciones, habrá que incorporar, al diagrama de bloques dado, bloques adicionales cuya función de transferencia sea unitaria que, o bien tengan como entrada una de las variables de entrada del sistema global, o bien tengan como salida una de las variables de salida del sistema global.
- Asimismo, será necesario asignar un número positivo (identificador) a cada uno de los bloques del diagrama resultante tras el cumplimiento de la condición anteriormente indicada.

La sintaxis de la función connect es la siguiente:

```
sysbc = connect(sysba, q , input, output)
```

dónde:

- *`sysba`*: modelo del sistema MIMO resultante de la agrupación de los modelos de todos los bloques que intervienen en el diagrama considerado. Para su obtención podrá hacerse uso de la función **append**. El orden que debe darse a estos modelos debe coincidir con el de la numeración creciente asignada a sus correspondientes bloques.
- *`input`*: vector cuyas componentes son los índices asignados a los bloques que tienen como entrada a una de las variables de entrada del sistema global.
- *`output`*: vector cuyas componentes son los índices asignados a los bloques que tienen como salida a una de las variables de salida del sistema global.
- *`q`*: matriz de conexión que especifica en su primer elemento de cada fila una entrada de algún sistema de la matriz *`sysba`* seguido de las salidas interconectadas a él bien sea directamente o a través de un bloque de suma. Las salidas pueden afectarse por un signo + o - de acuerdo al signo con que entran al bloque de suma. En caso de ser necesario, las filas irán rellenas con ceros para que el número de elementos de cada una de ellas sea el mismo. Por ejemplo, la fila $[1\ \ 2\ \ 0\ \ 0]$indica que el bloque identificado con el número está conectado a la salida del bloque 2, y la fila $[2\ \ 3\ \ 4\ \ -1]$indica que el bloque 2 está conectado a la salida del sumador: salida 3 + salida 4 - salida 1. En algunos casos puede surgir la necesidad de definir bloques unitarios ficticios para evitar que el índice de un determinado bloque aparezca más de una vez en la fila por existir más de una conexión con él.

El resultado de este comando es el modelo global *`sysbc`*, el cual viene expresado en ecuaciones en el espacio de estados de la forma:

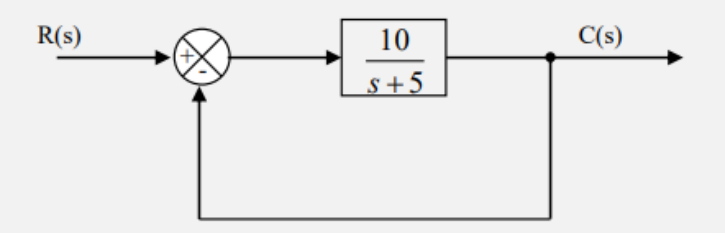
$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

En caso de que se deseara presentar con formato de función de transferencia el modelo global del sistema considerado, bastaría con aplicar, o bien la función **`tf`**, o bien la función **`zpk`**, al modelo determinado por el comando **`connect.`**

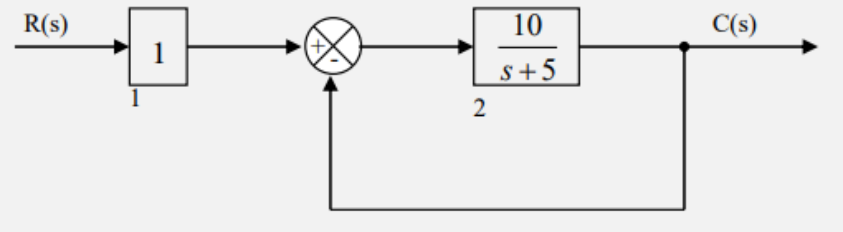
Contado de esta manera, parece un poco rompecabezas, ¿no? ¡Bajémoslo a la tierra con un ejemplito!

Ejemplo:

Reducir el siguiente diagrama de bloques:



Comenzaremos enumerando los bloques que aparecen en el diagrama y añadiendo los auxiliares necesarios:

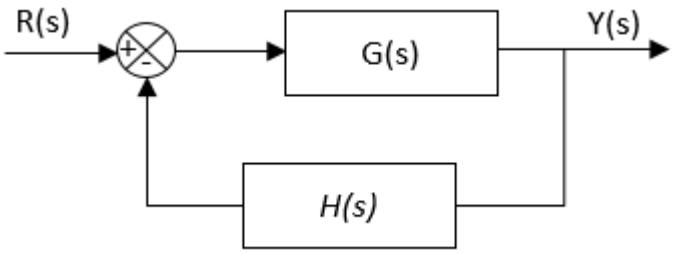


Nos metemos ya en el código...

```
G1 = 1; % Modelo del bloque 1
G2 = tf(10,[1 5]); % Modelo del bloque 2
sysba = append(G1,G2); % Modelo de sist. con entr. y salidas de bloques//// esta es
la forma que tiene matlab de saber que bloque hemos definido como uno y como dos

input = 1; % Indice correspondiente al bloque de entrada
output = 2; % Indice correspondiente al bloque de salida
q = [2 1 -2]; % Matriz de conexion entre bloques
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
G = minreal ( G )
```

```
G =

    10
  ------
  s + 15

Continuous-time transfer function.
Model Properties
```

Tarea 4: Calcule la función de transferencia en bucle cerrado de los sistemas realimentados (con realimentación negativa), esto es:



caracterizados por las siguientes funciones de transferencia de la planta, $G_i(s)$y del factor de realimentación, $H_i(s)$. Para ello habrán de ser empleados los siguientes procedimientos:

1. Analítico (a mano).
2. Puramente aritmético con modelos creados por MATLAB.
3. Mediante la función **`feedback`**.
4. Mediante las funciones **`append`** y **`connect`**

$$G_1(s) = \frac{s^2+5s+2}{s^3+3s^2+3s+1} \qquad H_1(s) = 1;$$

$$G_2(s) = \frac{7s^2+s+2}{s^3+s^2+1} \qquad H_2(s) = \frac{1}{s+1};$$

$$G_3(s) = \frac{s+1}{s^2+5s} \qquad H_3(s) = \frac{s+2}{s^2+4s+5}$$

```
% Tu código aquí
% G1
```

1º PURAMENTE CON MODELOS DE MATLAB

```
s = tf('s');
G1 = ((s^2)+5*s+2)/((s^3)+3*(s^2)+3*s+1);
H1 = 1;
H =zpk(G1/(1+G1*H1));
minreal(H)
```

```
ans =

        (s+4.562) (s+1)^3 (s+0.4384)
  ---------------------------------------
  (s+1)^3 (s+0.474) (s^2 + 3.526s + 6.329)

Continuous-time zero/pole/gain model.
Model Properties
```

```
%Si tachamos los dos terminos repetidos arriba y abajo nos queda la misma
%que usando el comando feedback
```

2º USANDO EL COMANDO FEEDBACK

```
H =feedback(G1, H1)
```

```
H =

      s^2 + 5 s + 2
  --------------------
  s^3 + 4 s^2 + 8 s + 3

Continuous-time transfer function.
Model Properties
```

```
H = zpk(H)
```

```
H =

        (s+4.562) (s+0.4384)
  --------------------------------
  (s+0.474) (s^2 + 3.526s + 6.329)

Continuous-time zero/pole/gain model.
Model Properties
```

3º MEDIANTE LAS FUNCIONES APPEND Y CONNECT

```
E = 1;
S = 1;
G1 = ((s^2)+5*s+2)/((s^3)+3*(s^2)+3*s+1); % Modelo del bloque 2
H1 = 1;  % Modelo del bloque 3(realimentación)
sysba = append(E,G1,H1,S); % Modelo de sist. con entr. y salidas de bloques////
esta es la forma que tiene matlab de saber que bloque hemos definido como uno y
como dos

input = 1; % Indice correspondiente al bloque de entrada
output = 2; % Indice correspondiente al bloque de salida
q = [2 1 -3;3 2 0;0 0 0]; % Matriz de conexion entre bloques
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
G = minreal ( G )
```

```
G =

      s^2 + 5 s + 2
  --------------------
  s^3 + 4 s^2 + 8 s + 3

Continuous-time transfer function.
Model Properties
```

Resultado esperado:

```
sys1_tf =

      s^2 + 5 s + 2
  ---------------------
  s^3 + 4 s^2 + 8 s + 3

Continuous-time transfer function.
```

```
% G2
```

1º PURAMENTE CON MODELOS DE MATLAB

```
s = tf('s');
G1 = ((7*s^2)+s+2)/((s^3)+(s^2)+1);
H1 = 1/(s+1);
H =tf(G1/(1+G1*H1));
minreal(H)
```

```
ans =

     7 s^3 + 8 s^2 + 3 s + 2
  ----------------------------
  s^4 + 2 s^3 + 8 s^2 + 2 s + 3

Continuous-time transfer function.
Model Properties
```

2º USANDO EL COMANDO FEEDBACK

```
H =feedback(G1, H1)
```

```
H =

     7 s^3 + 8 s^2 + 3 s + 2
  ----------------------------
  s^4 + 2 s^3 + 8 s^2 + 2 s + 3

Continuous-time transfer function.
Model Properties
```

3º MEDIANTE LAS FUNCIONES APPEND Y CONNECT

```
E = 1;
S = 1;
G1 = ((7*s^2)+s+2)/((s^3)+(s^2)+1);
H1 = 1/(s+1);
sysba = append(E,G1,H1,S);

input = 1; % Indice correspondiente al bloque de entrada
output = 2; % Indice correspondiente al bloque de salida
q = [2 1 -3;3 2 0;0 0 0]; % Matriz de conexion entre bloques
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
```

```
G = minreal ( G )
```

```
G =

     7 s^3 + 8 s^2 + 3 s + 2
  ----------------------------
  s^4 + 2 s^3 + 8 s^2 + 2 s + 3

Continuous-time transfer function.
Model Properties
```

Resultado esperado:

```
sys2_tf =

     7 s^3 + 8 s^2 + 3 s + 2
  ----------------------------
  s^4 + 2 s^3 + 8 s^2 + 2 s + 3

Continuous-time transfer function.
```

```
% G3
```

1º PURAMENTE CON MODELOS DE MATLAB

```
s = tf('s');
G1 = ((s+1)/((s^2)+5*s));
H1 = (s+2)/((s^2)+4*s+5);
H =tf(G1/(1+G1*H1));
minreal(H)
```

```
ans =

       s^3 + 5 s^2 + 9 s + 5
  -------------------------------
  s^4 + 9 s^3 + 26 s^2 + 28 s + 2

Continuous-time transfer function.
Model Properties
```

2º USANDO EL COMANDO FEEDBACK

```
H =feedback(G1, H1)
```

```
H =

       s^3 + 5 s^2 + 9 s + 5
  -------------------------------
  s^4 + 9 s^3 + 26 s^2 + 28 s + 2

Continuous-time transfer function.
Model Properties
```

3º MEDIANTE LAS FUNCIONES APPEND Y CONNECT

```
E = 1;
S = 1;
```

```
G1 = ((s+1)/((s^2)+5*s));
H1 = (s+2)/((s^2)+4*s+5);
sysba = append(E,G1,H1,S);

input = 1; % Indice correspondiente al bloque de entrada
output = 2; % Indice correspondiente al bloque de salida
q = [2 1 -3;3 2 0;0 0 0]; % Matriz de conexion entre bloques
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
G = minreal ( G )
```

```
G =

       s^3 + 5 s^2 + 9 s + 5
  -------------------------------
  s^4 + 9 s^3 + 26 s^2 + 28 s + 2

Continuous-time transfer function.
Model Properties
```

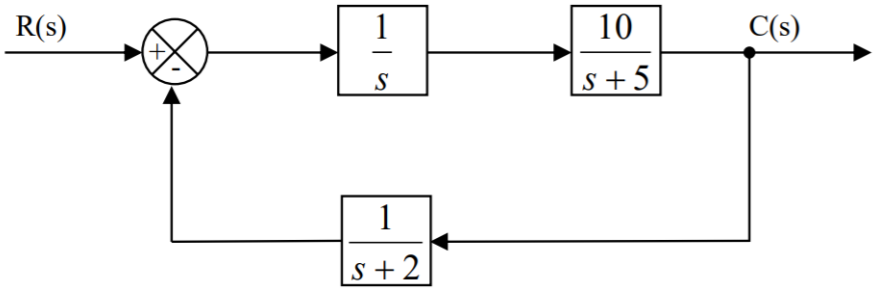
Resultado esperado:

```
sys3_tf =

       s^3 + 5 s^2 + 9 s + 5
  -------------------------------
  s^4 + 9 s^3 + 26 s^2 + 28 s + 2

Continuous-time transfer function.
```

Tarea 5: En las siguientes tareas, calcule la función de transferencia global de los sistemas realimentados caracterizados por los diagramas de bloques dados. Para ello habrán de ser empleados los siguientes procedimientos:

1. Analítico (mediante técnicas basadas en el álgebra de bloques).
2. Mediante las funciones **`append`** y **`connect`**.

Pista: Como puede verse, junto a cada diagrama de bloques se muestra la correspondiente solución.



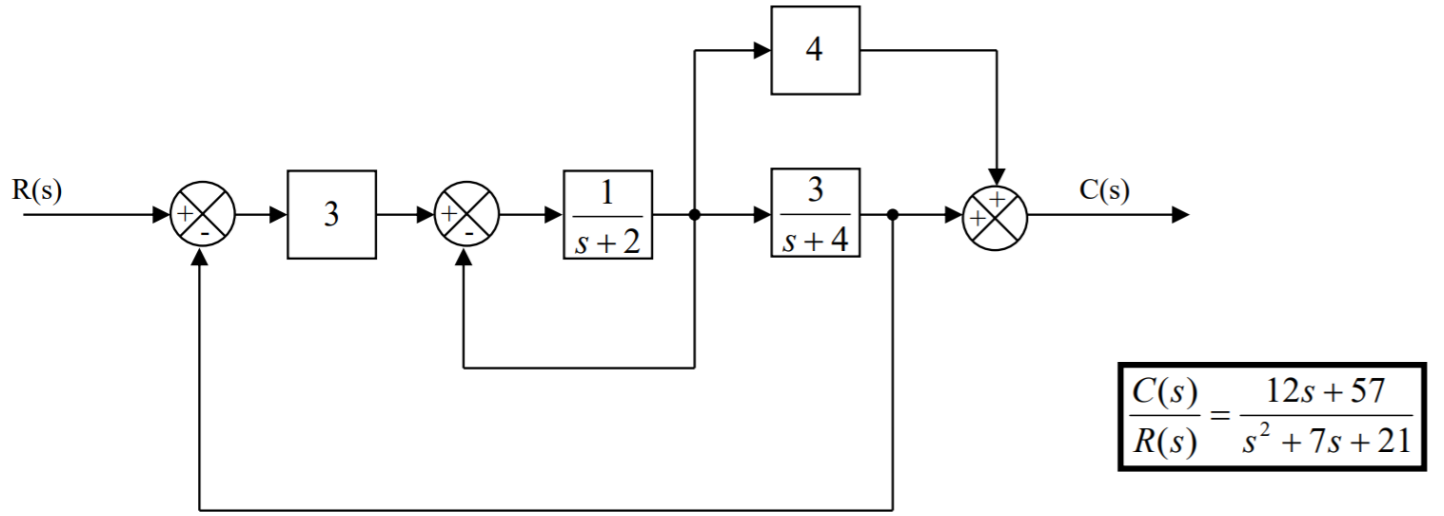
$$\frac{C(s)}{R(s)} = \frac{10s+20}{s^3+7s^2+10s+10}$$

```
% Tu código aquí
s = tf('s');
E = 1;
```

```
S = 1;
G1 = (10/((s^2)+5*s));
H1 = 1/((s+2));
sysba = append(E,G1,H1,S);

input = 1; % Indice correspondiente al bloque de entrada
output = 2; % Indice correspondiente al bloque de salida
q = [2 1 -3;3 2 0;4 2 0]; % Matriz de conexion entre bloques
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
G = minreal ( G )
```

```
G =

        10 s + 20
  ----------------------
  s^3 + 7 s^2 + 10 s + 10

Continuous-time transfer function.
Model Properties
```

Tarea 6:

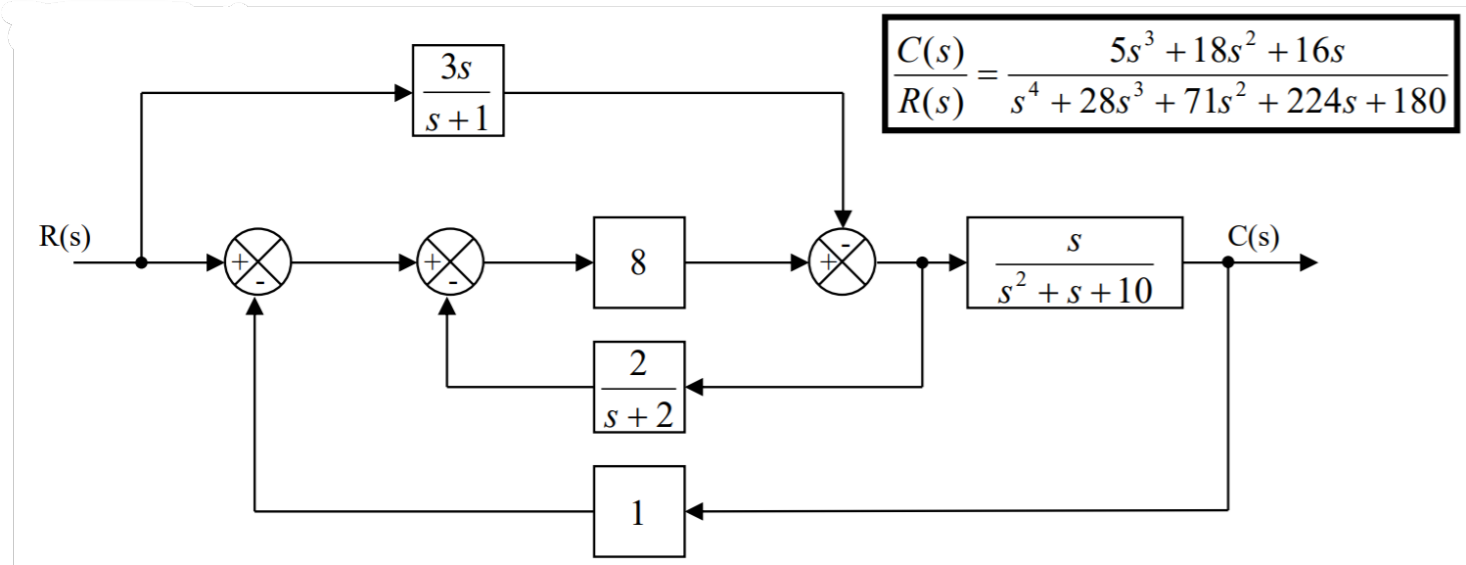



```
% Tu código aquí
s = tf('s');
E = 1;
S = 1;
G1 = 3;
G2 = 1/(s+2);
G3 = 3/(s+4);
H1 = 4;
sysba = append(E,G1,G2,G3,H1,S);
```

```
input = 1; % Indice correspondiente al bloque de entrada
output = 6; % Indice correspondiente al bloque de salida
q = [2 1 -4 0;3 2 -3 0;4 3 0 0;5 3 0 0;6 4 5 0]; % Matriz de conexion entre bloques
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
G = minreal ( G )
```

```
G =

    12 s + 57
  --------------
  s^2 + 7 s + 21

Continuous-time transfer function.
Model Properties
```

Tarea 7:

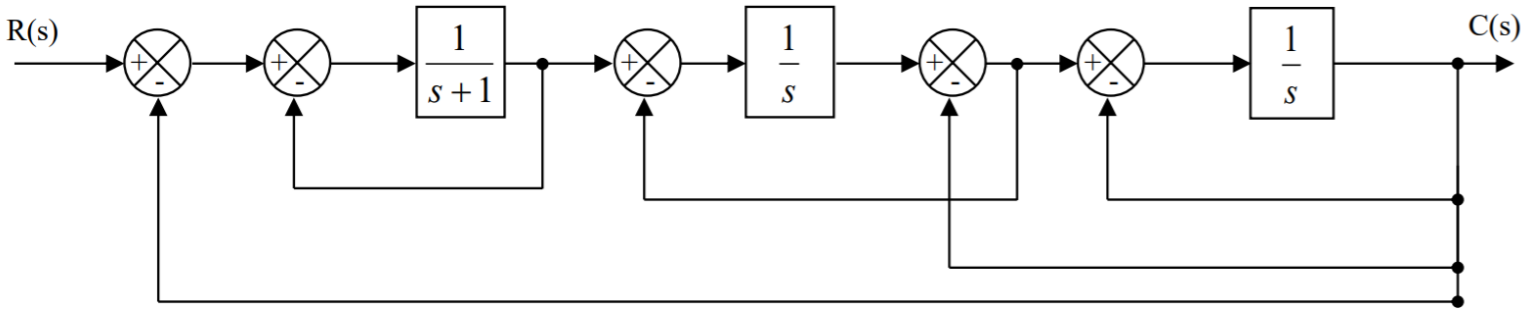



```
% Tu código aquí
s = tf('s');
E = 1;
S = 1;
G1 = 1;
G2 = 8;
G3 = s/((s^2)+s+10);
H3 = 2/(s+2);
H2 = 1;
H1 = 3*s/(s+1);
sysba = append(E,G1,H1,G2,G3,H3,H2,S);

input = 1; % Indice correspondiente al bloque de entrada
output = 8; % Indice correspondiente al bloque de salida
q=[2 1 -7; 3 1 0; 4 2 -6; 5 4 -3; 6 4 -3; 7 5 0;8 5 0];
```

```
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
G = minreal ( G )
```

```
G =

   5 s^3 + 18 s^2 + 16 s + 6.161e-16
  ----------------------------------
  s^4 + 28 s^3 + 71 s^2 + 224 s + 180

Continuous-time transfer function.
Model Properties
```

Tarea 8:



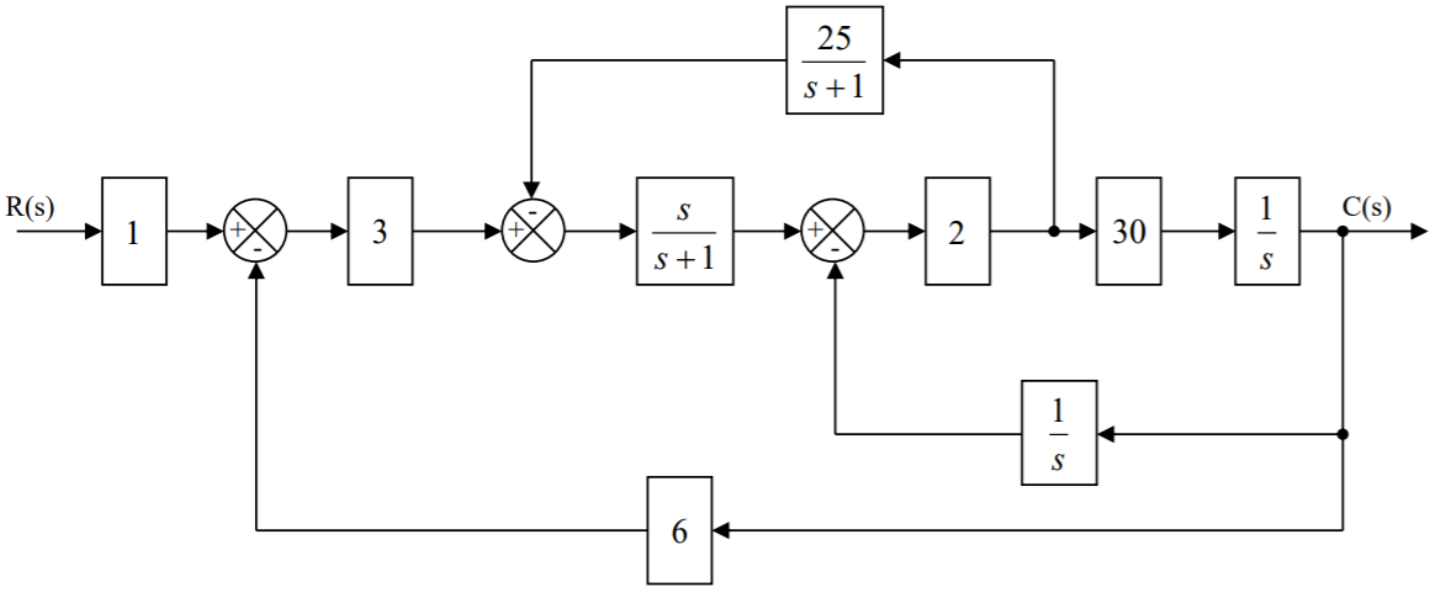
$$\frac{C(s)}{R(s)} = \frac{1}{s^3 + 5s^2 + 7s + 3}$$

```
% Tu código aquí
s = tf('s');
E = 1;
S = 1;
G1 = 1;
G2 = 1/(s+1);
G3 = 1/s;
G4 = 1;
G5 = 1/s;
sysba = append(E,G1,G2,G3,G4,G5,S);

input = 1; % Indice correspondiente al bloque de entrada
output = 7; % Indice correspondiente al bloque de salida
q=[2 1 -6; 3 2 -3; 4 3 -5; 5 4 -6; 6 5 -6; 7 6 0];
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
```

```
G = minreal ( G )
```

```
G =

           1
  --------------------
  s^3 + 5 s^2 + 7 s + 3

Continuous-time transfer function.
Model Properties
```

Tarea 9:



$$\frac{C(s)}{R(s)} = \frac{180s^3 + 180s^2}{s^4 + 1132s^3 + 1141s^2 + 120s + 60}$$

```
% Tu código aquí
s = tf('s');
E = 1;
S = 1;
G2 = 1;
G3 = 3;
G4 = s/(s+1);
G5 = 2;
G6 = 30;
G7 = 1/s;
H8 = 25/(s+1);
H9 = 1/s;
```

```
H10 = 6;
sysba = append(E,G2,G3,G4,G5,G6,G7,H8,H9,H10,S);

input = 1; % Indice correspondiente al bloque de entrada
output = 11; % Indice correspondiente al bloque de salida
q=[2 1 0; 3 2 -10; 4 3 -8; 5 4 -9; 6 5 0; 7 6 0; 8 5 0; 9 7 0; 10 7 0; 11 7 0];
sysb_ss = connect(sysba,q,input,output); % Modelo global en ss.
G = tf(sysb_ss); % Modelo global equivalente en formato tf
% Modelo global simplificado (definitivo) con formato tf
G = minreal ( G )
```

```
G =

  180 s^3 + 180 s^2 + 1.385e-12 s + 6.049e-30
  -------------------------------------------
    s^4 + 1132 s^3 + 1141 s^2 + 120 s + 60

Continuous-time transfer function.
Model Properties
```

Simulación de sistemas

O de como se comporta mi sistema ante distintas entradas

Table of Contents

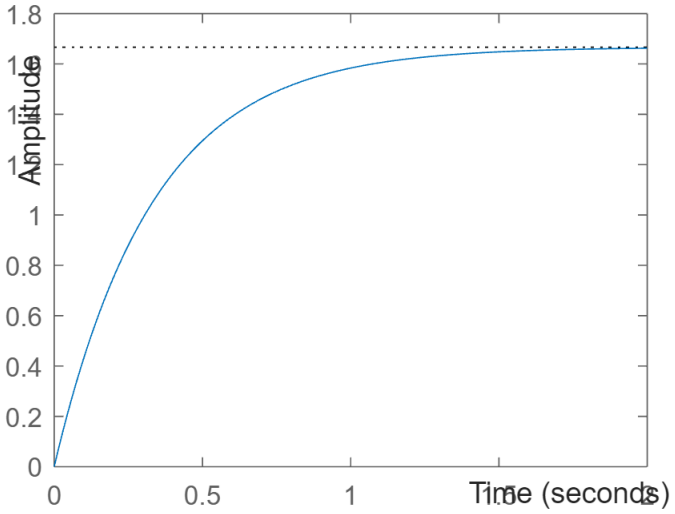
Venimos acumulando experiencia en cuanto al trabajo con la descripción externa (comandos **`tf`**, **`zpk`**, forma directa) y descripción interna (comando **`ss`**) de sistemas, o lo que es lo mismo, la representación mediante modelos matemáticos de dichos sistemas. En este cuaderno vamos tratar las posibilidades que ofrece MATLAB para simular sistemas, esto es, obtener la evolución de su respuesta ante ciertas entradas de interés.

1. La entrada estrella: el escalón

El comando **`step`** nos permite obtener la respuesta de un sistema ante una entrada escalón. Su sintaxis es la que sigue:

- *`step(sys):`* Obtiene en una nueva ventana la representación gráfica de la respuesta que presentaría, ante la aplicación de una señal de entrada escalón unitario, un sistema cuyo modelo matemático (ya sea en descripción externa o interna) fuese el almacenado en la variable *`sys`*. El modelo considerado puede ser de tiempo continuo o de tiempo discreto y SISO o MIMO. En el caso de un sistema de múltiples entradas, obtiene en la misma ventana un conjunto de gráficas asociado a las respuestas que el mismo presentaría, ante la aplicación de una señal escalón unitario, en cada uno de sus canales de entrada por separado. Ejemplo:

```
s = tf('s');
G = 5/(s+3); % Definimos la función de transferencia
step(G)      % Obtenemos su respuesta ante entrada escalón unitario
```



Nótese que esto sería equivalente a solucionar la ecuación diferencial que dio origen a dicha función de transferencia empleando la transformada inversa de Laplace y evaluarla en los instantes de tiempo de interés. Con Matlab:

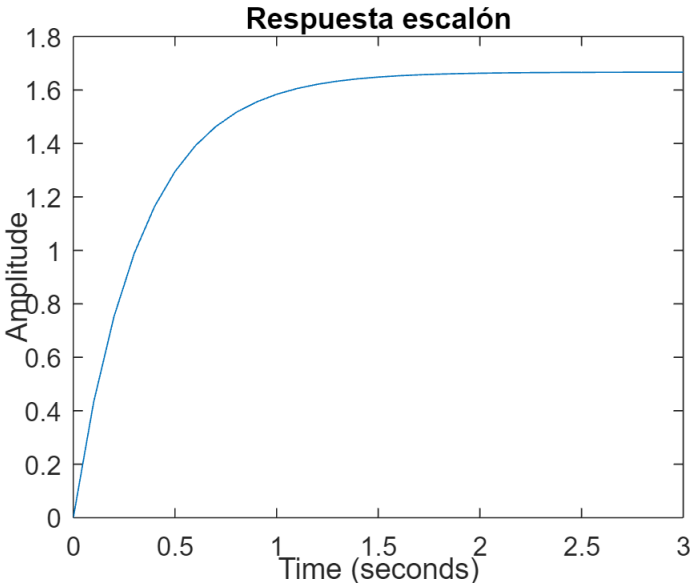
```
syms s t
F = (5/(s+3))*1/s;
res = ilaplace(F)
```

```
res =
```

$$\frac{5}{3} - \frac{5\,\mathrm{e}^{-3t}}{3}$$

O lo que es lo mismo, $y(t) = 1.6667 - 1.6667e^{-3t}$. Si evaluamos esta función en el mismo rango temporal que la simulación anterior, oh maravilla: obtendremos la misma respuesta:

```
x = 0:0.1:3;
y = zeros(length(x),1);
for i=1:length(x)
    y(i) = subs(res,t,x(i));
end
plot(x,y), title('Respuesta escalón'), xlabel('Time (seconds)'), ylabel('Amplitude')
```



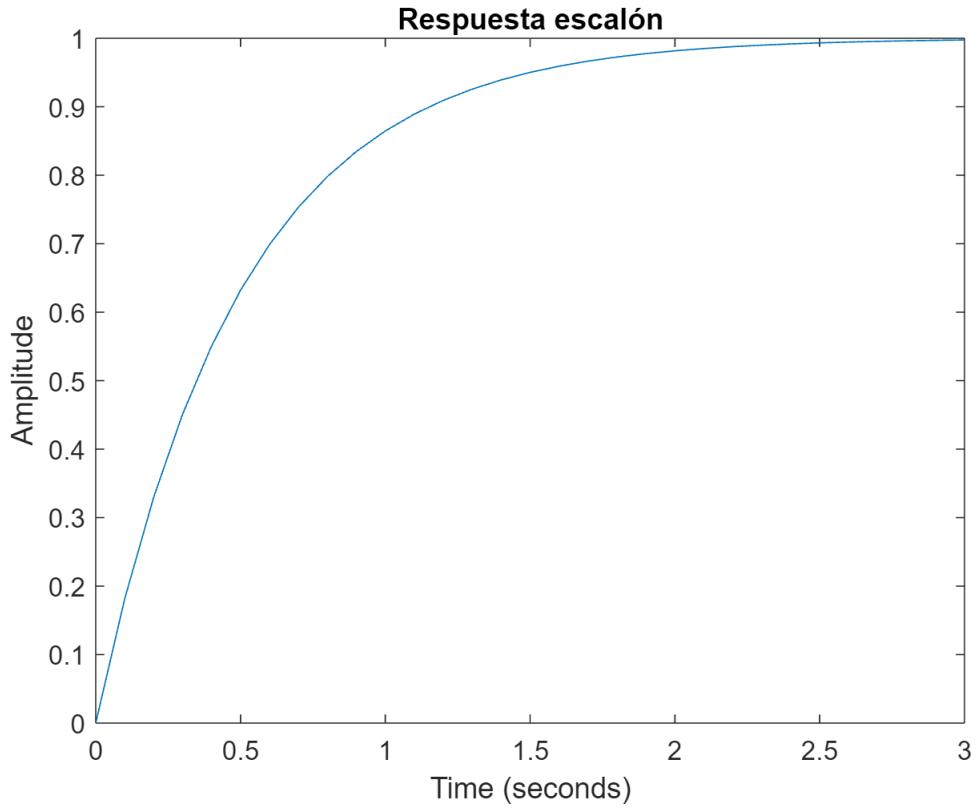
Tarea 1: Analice, de forma cualitativa (estabilidad, ganancia, etc.), cual sería la respuesta esperable, ante una señal de entrada escalón unitario, de los sistemas cuya función de transferencia se indica y verifíquelo empleando el comando **`step`**.

$G_1(s) = \frac{2}{(s+2)}$ (un polo en el semiplano izquierdo)

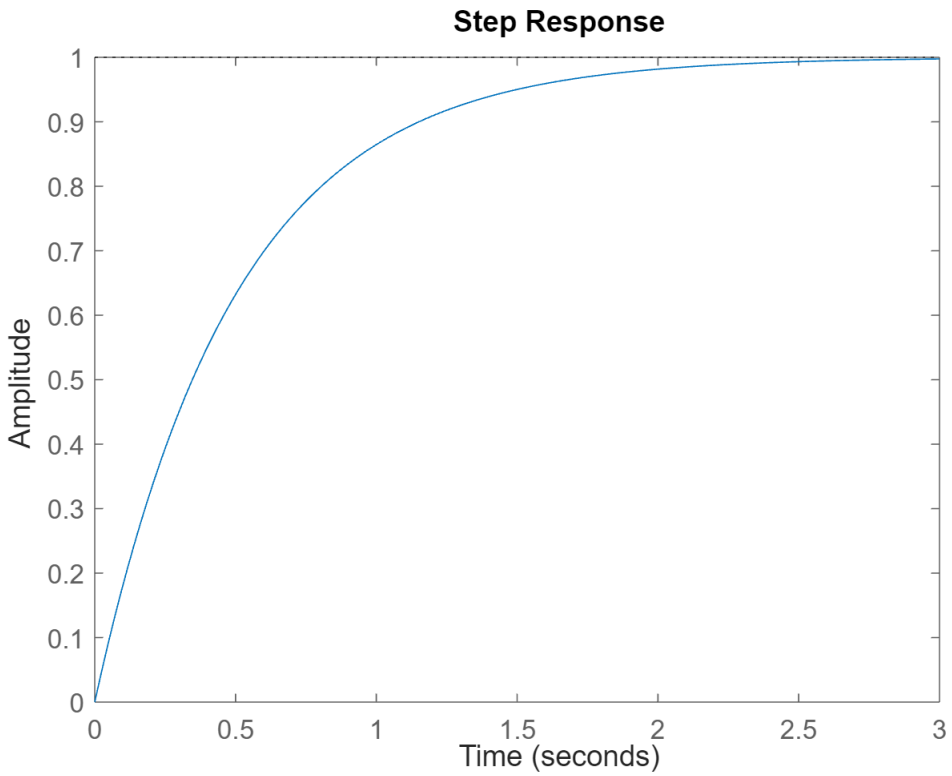
```
% Tu código aquí:
syms s t
G1 = (2/(s+2))*(1/s);
res1 = ilaplace(G1)
```

```
res1 = 1 - e^(-2*t)
```

```
x = 0:0.1:3;
y = zeros(length(x),1);
for i=1:length(x)
    y(i) = subs(res1,t,x(i));
end
plot(x,y), title('Respuesta escalón'), xlabel('Time (seconds)'), ylabel('Amplitude')
```




```
s = tf('s');
G1 = 2/(s+2);
step(G1)
```



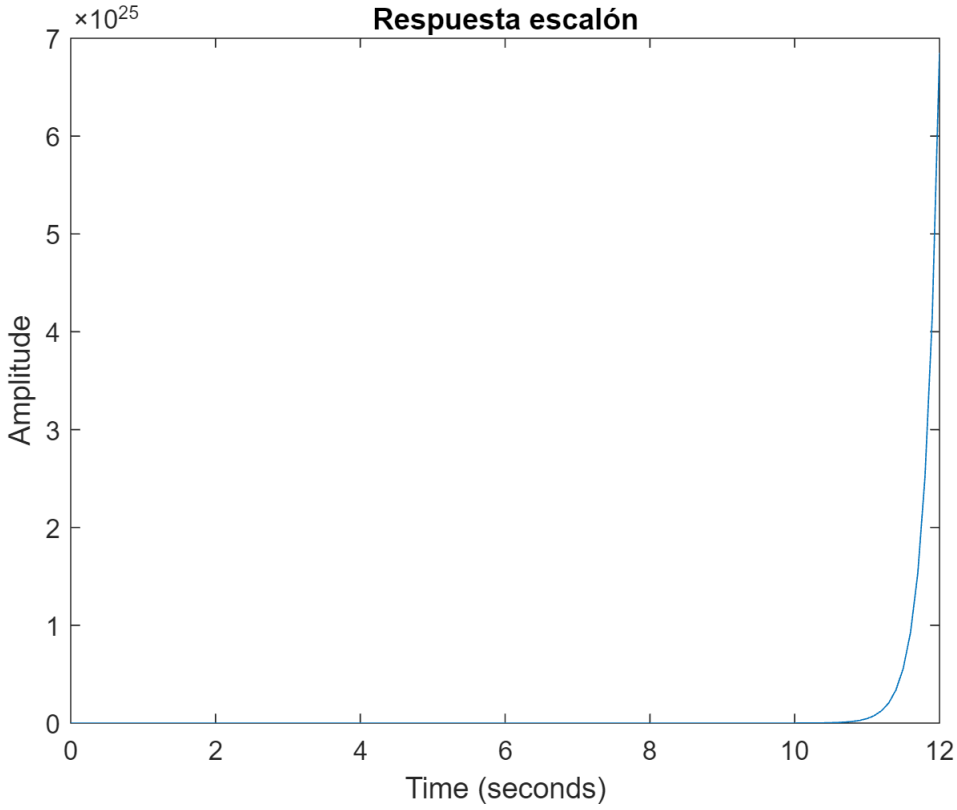
$G_2(s) = \frac{3}{(s-5)}$ (un polo en el semiplano derecho)

```
% Tu código aquí:
syms s t
G2 = (3/(s-5))*(1/s);
res2 = ilaplace(G2)
```

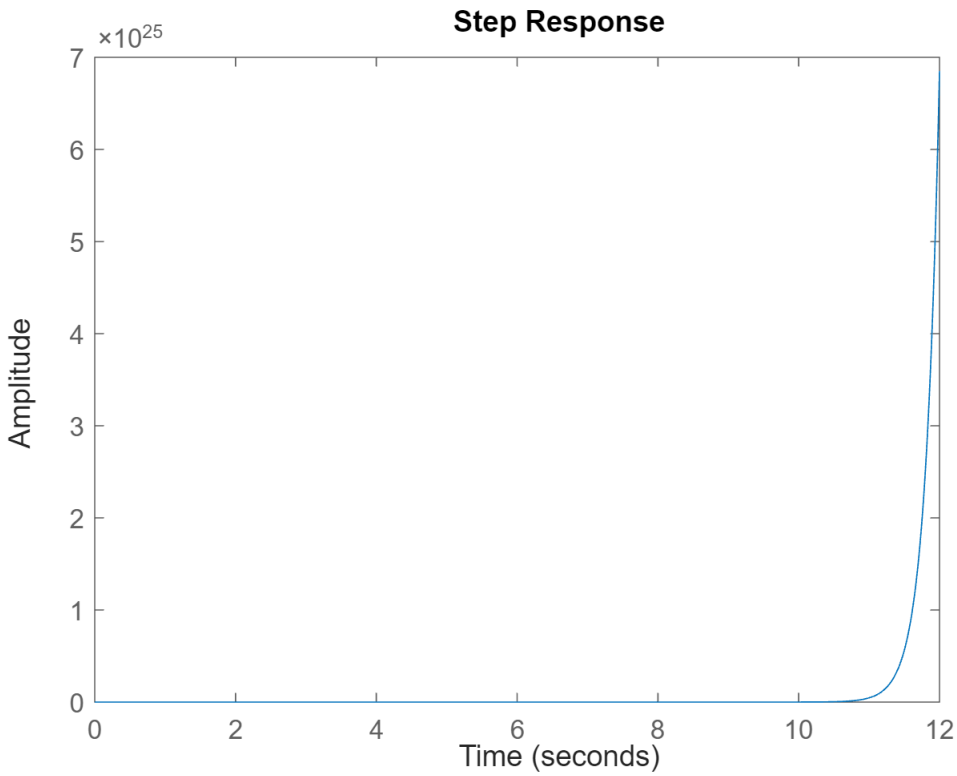
```
res2 =
```

$$\frac{3\,e^{5t}}{5} - \frac{3}{5}$$

```
x = 0:0.1:12;
y = zeros(length(x),1);
for i=1:length(x)
    y(i) = subs(res2,t,x(i));
end
plot(x,y), title('Respuesta escalón'), xlabel('Time (seconds)'), ylabel('Amplitude')
```




```
s = tf('s');
G2 = 3/(s-5);
step(G2)
```



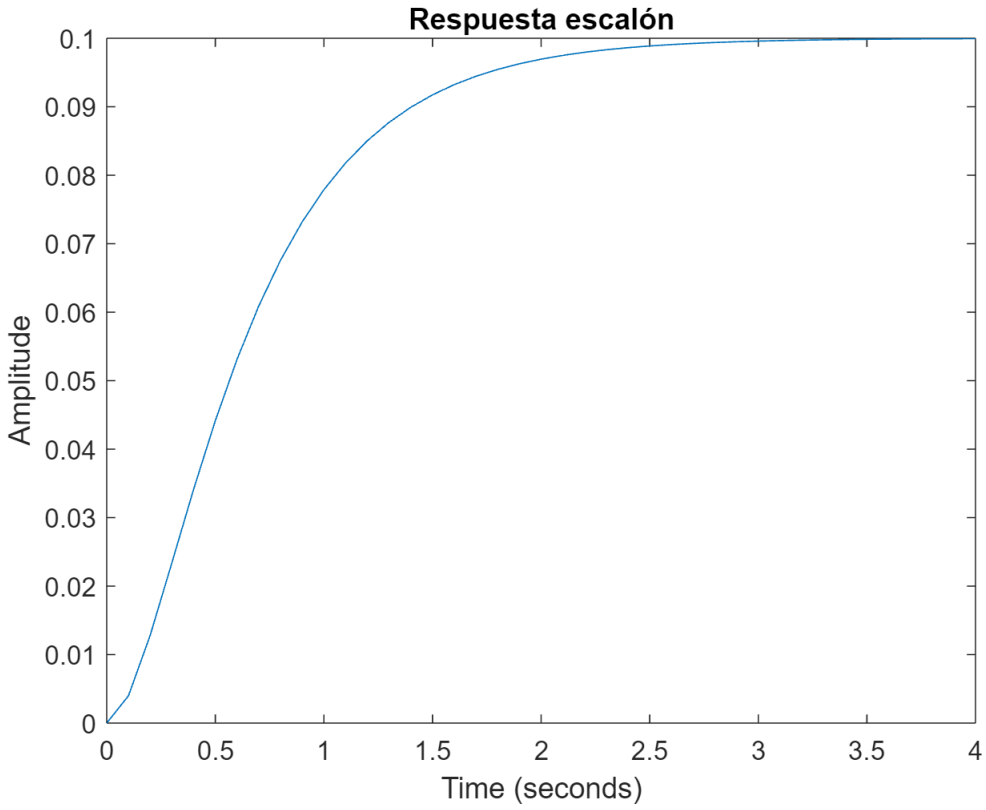
$G_3(s) = \frac{1}{(s+2)(s+5)}$ (dos polos reales en el semiplano izquierdo)

```
% Tu código aquí:
syms s t
G3 = (1/((s+2)*(s+5)))*(1/s);
res3 = ilaplace(G3)
```

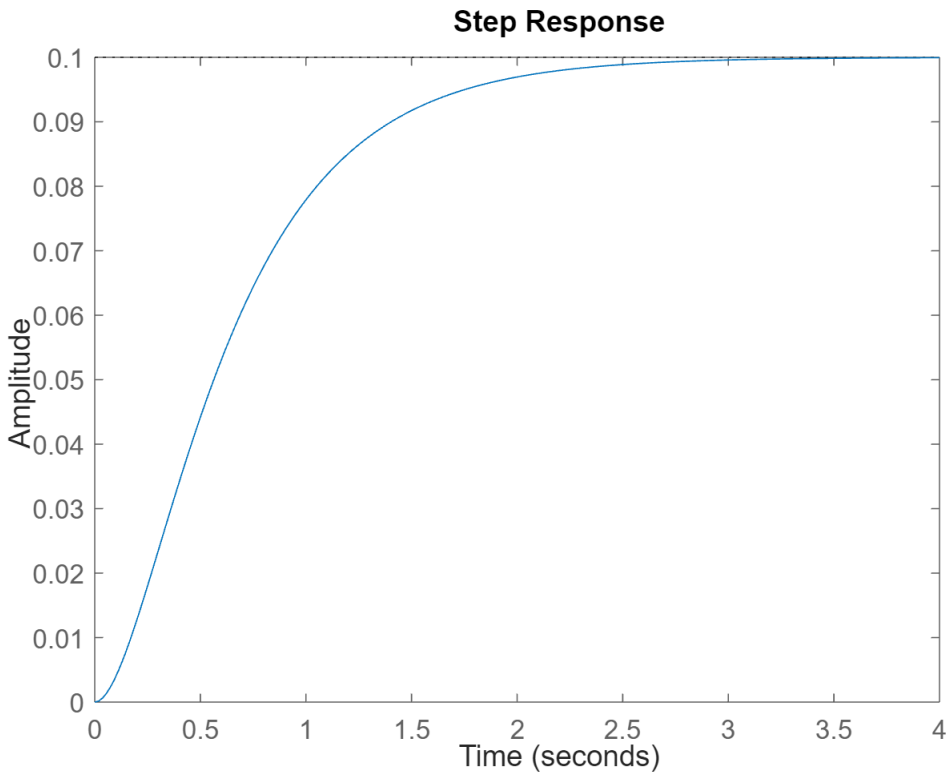
```
res3 =
```

$$\frac{e^{-5t}}{15} - \frac{e^{-2t}}{6} + \frac{1}{10}$$

```
x = 0:0.1:4;
y = zeros(length(x),1);
for i=1:length(x)
    y(i) = subs(res3,t,x(i));
end
plot(x,y), title('Respuesta escalón'), xlabel('Time (seconds)'), ylabel('Amplitude')
```




```
s = tf('s');
G3 = (1/((s+2)*(s+5)));
step(G3)
```



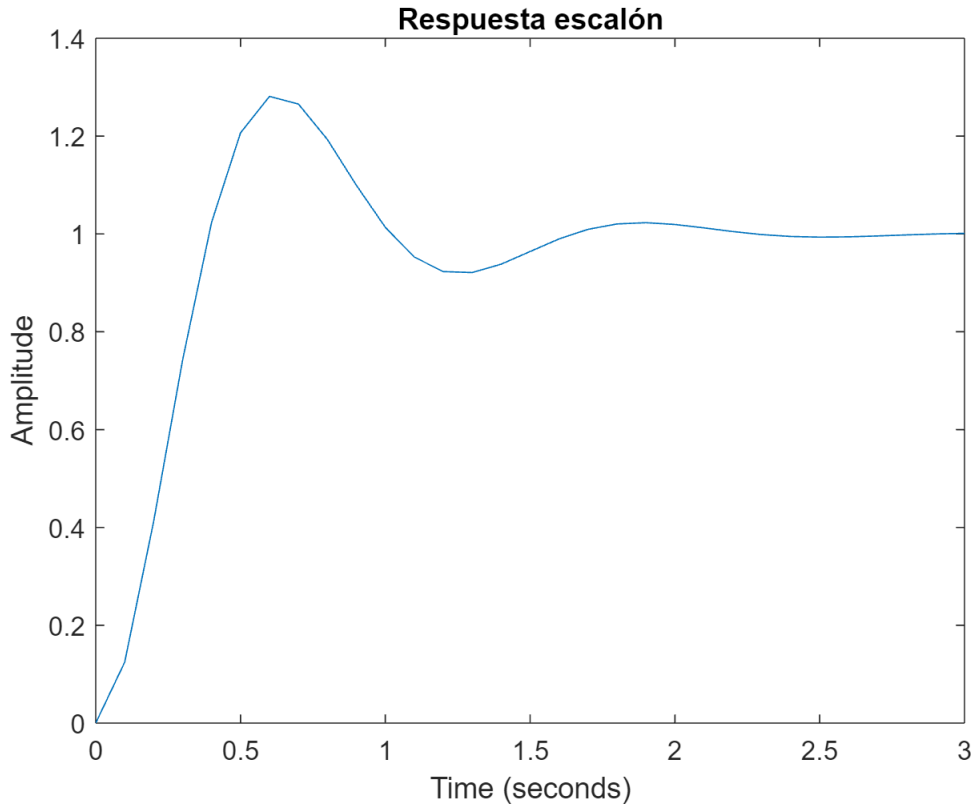
$G_4(s) = \frac{29}{(s+2+5j)(s+2-5j)}$ (dos polos complejos en el semiplano izquierdo)

```
% Tu código aquí:
syms s t
G4 = (29/((s+2+5j)*(s+2-5j)))*(1/s);
res4 = ilaplace(G4)
```

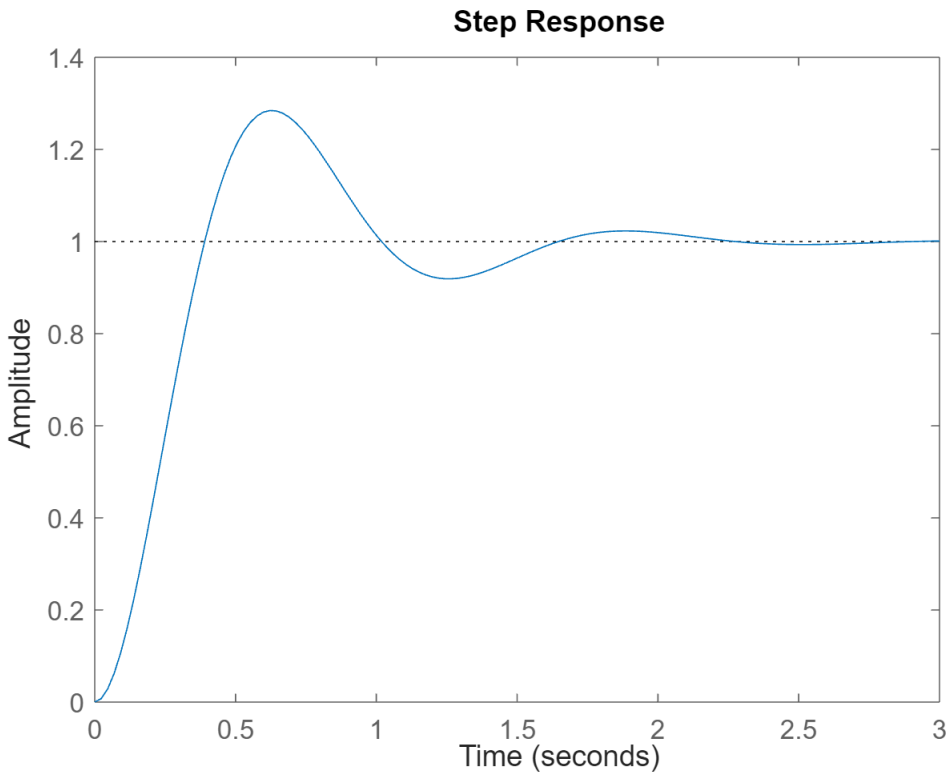
```
res4 =
```

$$1 + e^{t\,(-2-5\,i)}\left(-\frac{1}{2}-\frac{1}{5}\,i\right) + e^{t\,(-2+5\,i)}\left(-\frac{1}{2}+\frac{1}{5}\,i\right)$$

```
x = 0:0.1:3;
y = zeros(length(x),1);
for i=1:length(x)
    y(i) = subs(res4,t,x(i));
end
plot(x,y), title('Respuesta escalón'), xlabel('Time (seconds)'), ylabel('Amplitude')
```




```
s = tf('s');
G4 = 29/((s+2+5j)*(s+2-5j));
step(G4)
```



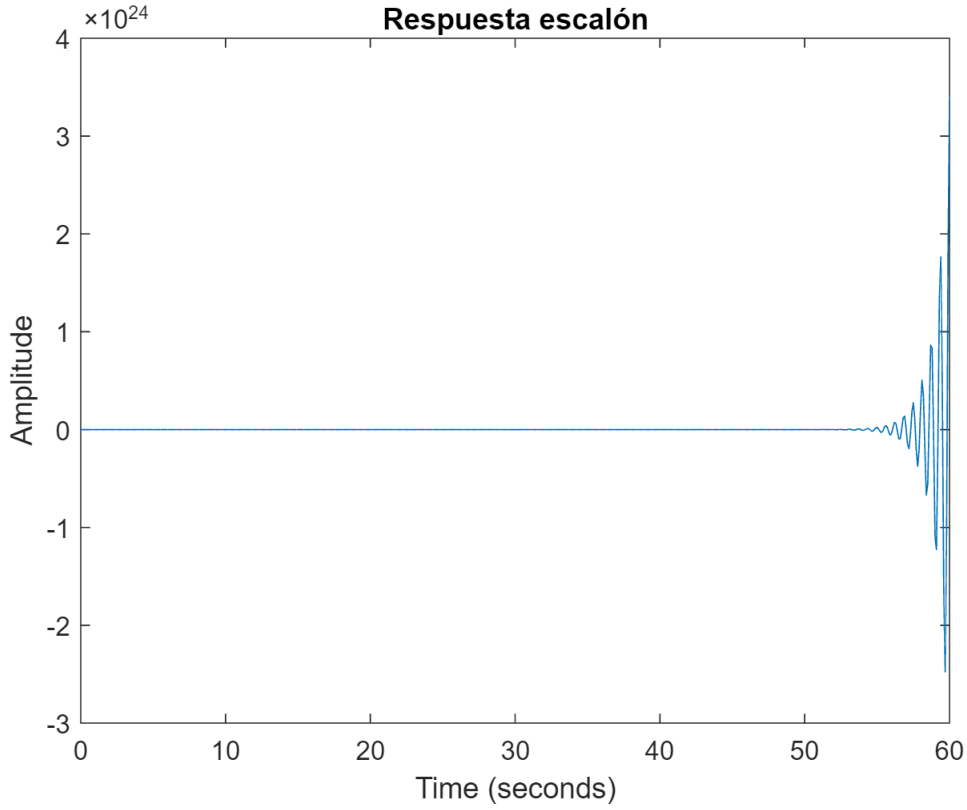
$$G_5(s) = \frac{3}{(s-1+10j)(s-1-10j)}$$

```
% Tu código aquí:
syms s t
G5 = (3/((s-1+10j)*(s-1-10j)))*(1/s);
res5 = ilaplace(G5)
```

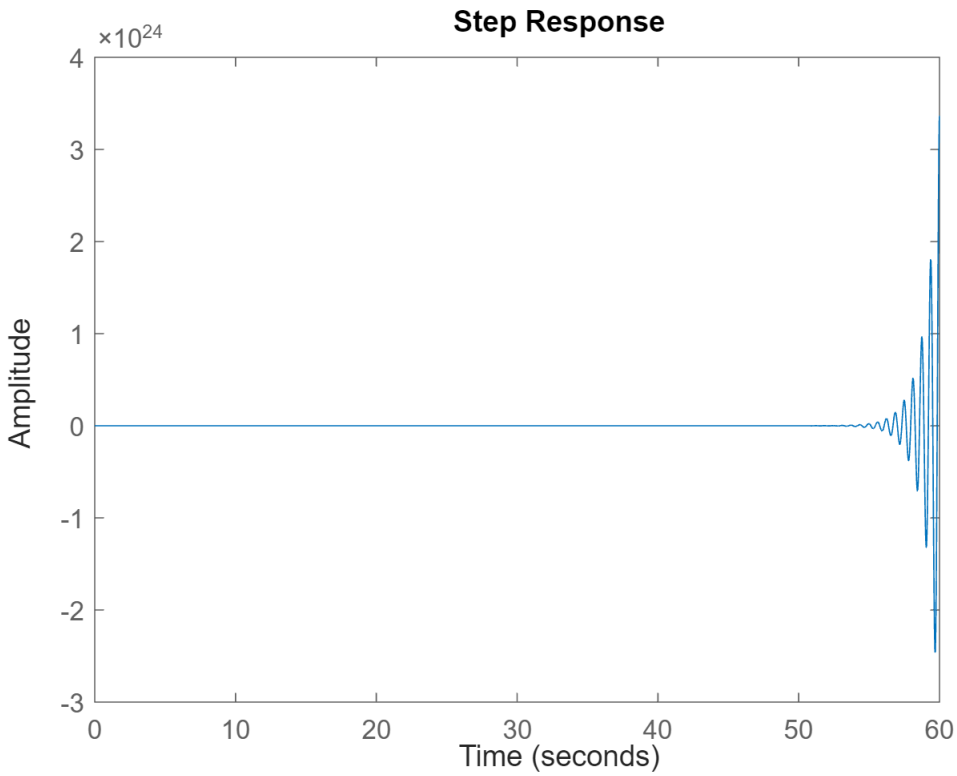
```
res5 =
```

$$\frac{3}{101} + e^{t\,(1-10\,i)}\left(-\frac{3}{202} + \frac{3}{2020}\,i\right) + e^{t\,(1+10\,i)}\left(-\frac{3}{202} - \frac{3}{2020}\,i\right)$$

```
x = 0:0.1:60;
y = zeros(length(x),1);
for i=1:length(x)
    y(i) = subs(res5,t,x(i));
end
plot(x,y), title('Respuesta escalón'), xlabel('Time (seconds)'), ylabel('Amplitude')
```



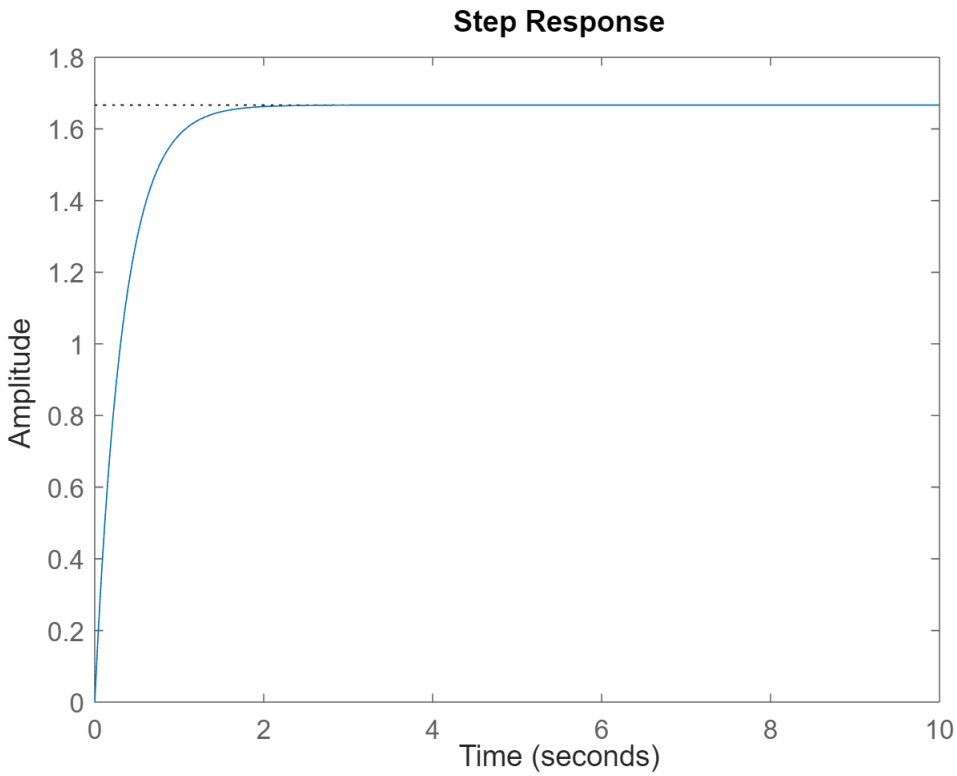

```
s = tf('s');
G5 = (3/((s-1+10j)*(s-1-10j)));
step(G5)
```




```

```

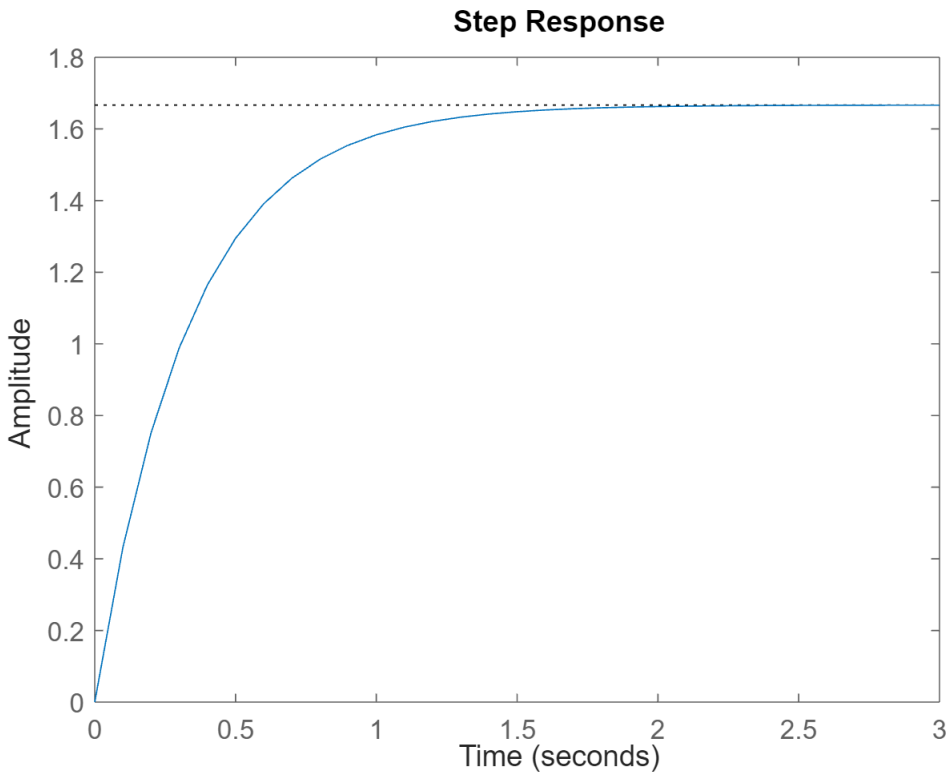
- *step(sys, tfinal):* Obtiene un resultado análogo al del caso anterior. La respuesta se representará hasta un instante de tiempo igual al valor de tfinal expresado en segundos. Ejemplo:

```
tfinal = 10;   % Definimos el instante de tiempo en el que va a finalizar la
visualización de la respuesta
step(G,tfinal) % Obtenemos su respuesta ante entrada escalón unitario hasta tfinal
```



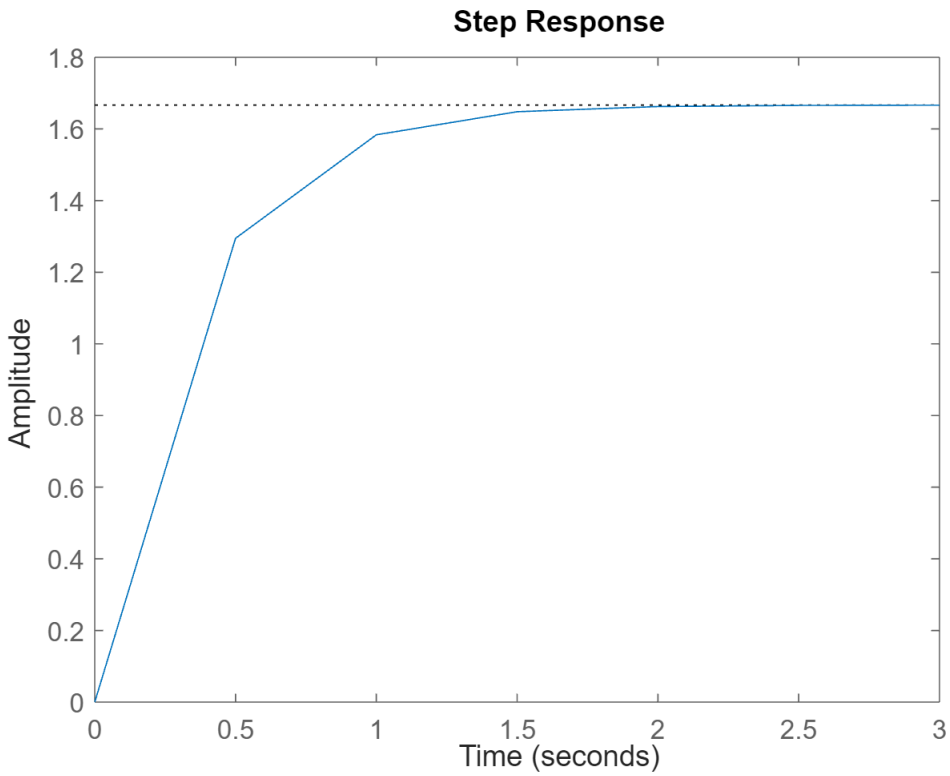
- *step(sys, t):* Obtiene un resultado análogo al de los casos anteriores. La respuesta se representará para un vector de tiempos dado por el valor de *t*. Éste debe ser un vector de elementos igualmente espaciados (*dt* segundos), de la forma: *t = 0:dt:tfinal*. En cualquier caso, dicha separación *dt* deberá ser escogida de forma que la respuesta transitoria sea visible con suficiente detalle. Ejemplo:

```
t = 0:0.1:3;  % Definimos el vector de elementos igualmente espaciados cada 0.1s
step(G,t)     % Obtenemos la respuesta escalón en dichos instantes de tiempo
```



Ejemplo con un *dt* demasiado grande:

```
t = 0:0.5:3;  % Definimos el vector de elementos igualmente espaciados cada 0.5s
step(G,t)     % Obtenemos la respuesta escalón en dichos instantes de tiempo
```



1.1 Simulación simultánea de varios sistemas

MATLAB permite mostrar la respuesta de varios sistemas ante la misma entrada en un mismo gráfico. Para ello se puede usar el comando **`step`** con la siguiente sintáxis:

- *`step(sys1, sys2, sys3, …):`* Obtiene en una nueva ventana la representación gráfica de las respuestas que presentarían, ante la aplicación de una señal de entrada escalón unitario, los sistemas cuyos modelos matemáticos fuesen los almacenados en las variables *`sys1`*,*`sys2`*, *`sys3`*, etc. Esta forma de llamada a la función **`step`** es útil para comparar respuestas de múltiples sistemas. Ejemplo:

```
Gbis = 3/(s+2); % Definimos el modelo de un segundo sistema
step(G,Gbis) % Mostramos la respuesta escalón de ambos
```

```
Error using DynamicSystem/step
In time response commands, the final time must be a positive scalar.
```

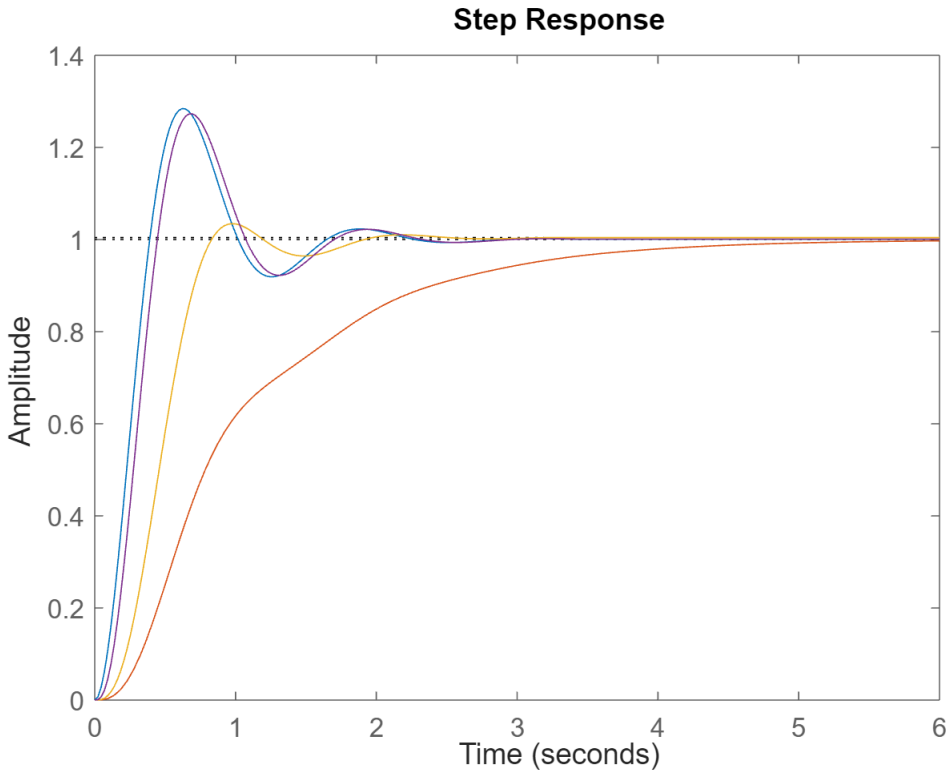
```
legend
```

Tarea 2: Para el sistema cuya función de transferencia es $G_4(s)$, indicar qué influencia tendría, en su respuesta temporal ante una señal de entrada escalón unitario, el añadido de un polo real situado en el semiplano izquierdo complejo. ¿Cómo afectaría la ubicación de dicho polo con respecto a la que tienen los polos de la función de transferencia original? Obtenga y compare las respuestas que presentaría ante una señal de entrada escalón unitario dicho sistema, con y sin el polo añadido, si éste es ubicado en -1, -3 y -20 (para ello

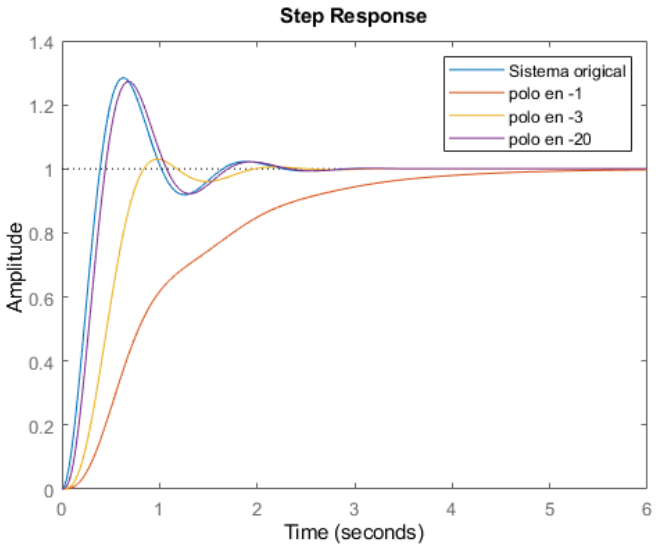
deberá superponer las gráficas de ambas respuestas, empleando, si es preciso a efectos de comparación, una ganancia que asegure que las mismas se estabilicen en 1).

Respuesta a la pregunta aquí:

```
% Tu código aquí
s = tf('s');
G4 = 29/((s+2+5j)*(s+2-5j));
G4_1 = (29/((s+2+5j)*(s+2-5j)))*(1/(s+1));
G4_3 = (29/((s+2+5j)*(s+2-5j)))*(1/(s+3));
G4_20 = (29/((s+2+5j)*(s+2-5j)))*(1/(s+20));
step(G4,(G4_1),(G4_3/0.332),(G4_20/0.05))
```

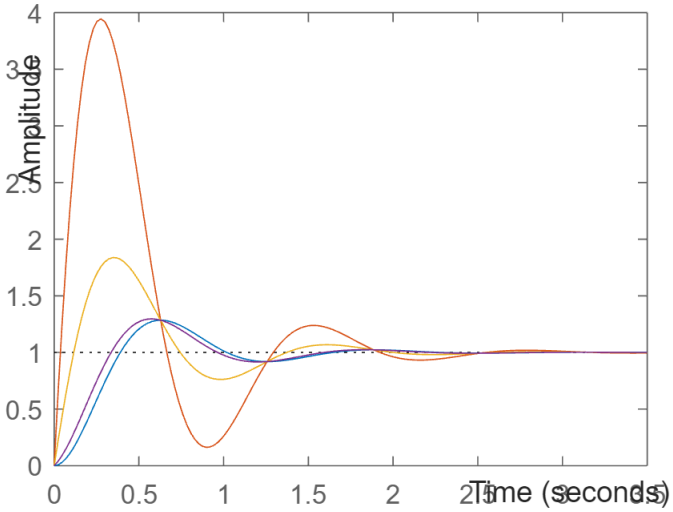


Resultado esperado:

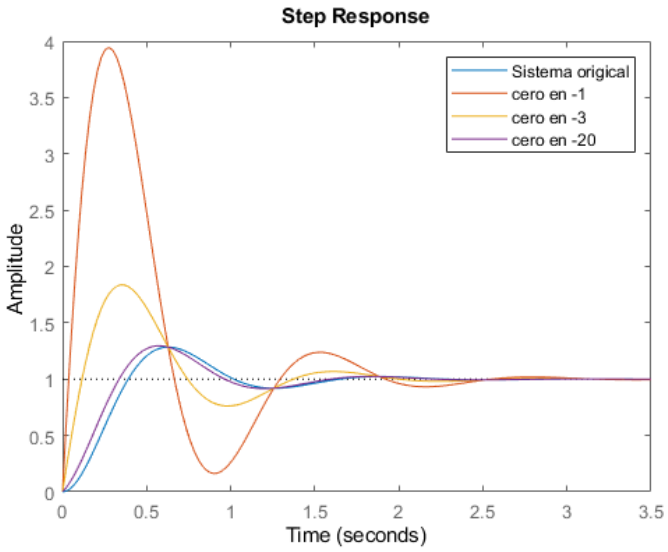


Tarea 3: Para el sistema cuya función de transferencia es $G_4(s)$, indicar qué influencia tendría, en su respuesta temporal ante una señal de entrada escalón unitario, el añadido de un cero real situado en el semiplano izquierdo complejo. ¿Cómo afectaría la ubicación de dicho cero con respecto a la que tienen los polos de la función de transferencia original? Obtenga y compare las respuestas que presentaría ante una señal de entrada escalón unitario dicho sistema, con y sin el cero añadido, si éste es ubicado en -1, -3 y -20 (superponer las correspondientes gráficas conforme a lo indicado en el ejercicio anterior).

```
% Tu código aquí
G4=zpk([],[-2+5*i -2-5*i],29);
G4_1=zpk(-1,[-2+5*i -2-5*i],29);
G4_3=zpk(-3,[-2+5*i -2-5*i],29/3);
G4_20=zpk(-20,[-2+5*i -2-5*i],29/20);
step(G4,G4_1,G4_3,G4_20)
```



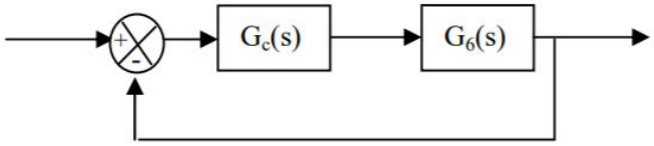
Resultado esperado:



Tarea 4: La función de transferencia $G_6(s)$ dada representa el modelo de una planta que es regulada, en bucle cerrado y con realimentación negativa y unitaria, mediante un controlador cuya función de transferencia, $G_c(s)$, es igualmente dada. Obtenga la respuesta temporal que este sistema realimentado presentaría ante una señal de entrada escalón unitario para los siguientes valores de la ganancia K indicada en el modelo del controlador: $K = 1$, $K = 4$, $K = 8.4$ y $K = 15$. ¿Qué conclusiones pueden alcanzarse a partir de los resultados obtenidos?

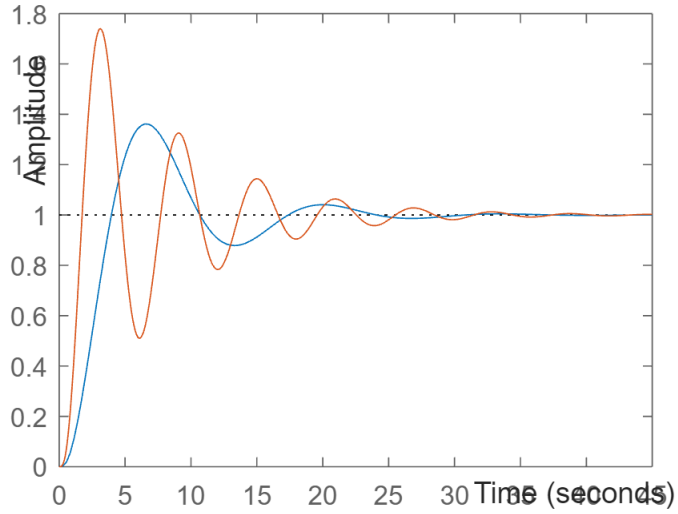
$$G_c(s) = K \cdot \frac{(s+1)}{s \cdot (s+5)}$$

$$G_6(s) = \frac{20}{s^3 + 12s^2 + 21s + 5}$$

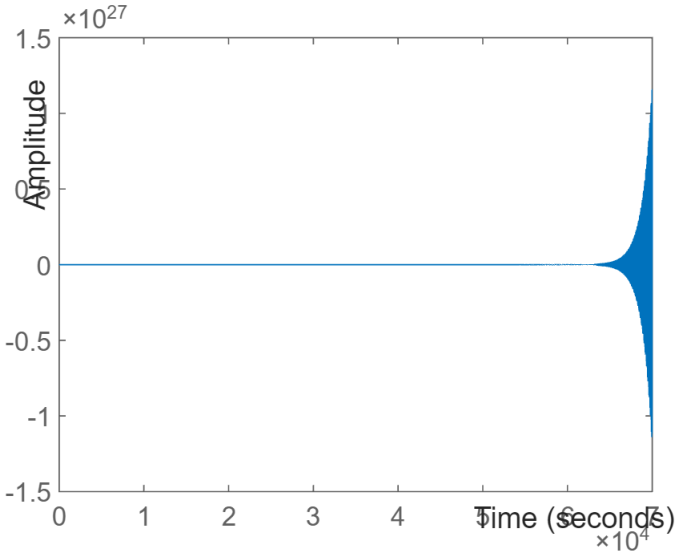


Pista: Muestra los resultados para *K=1* y *K=4* en una gráfica, y para *K=8.4* y *K=15* en otra distinta.

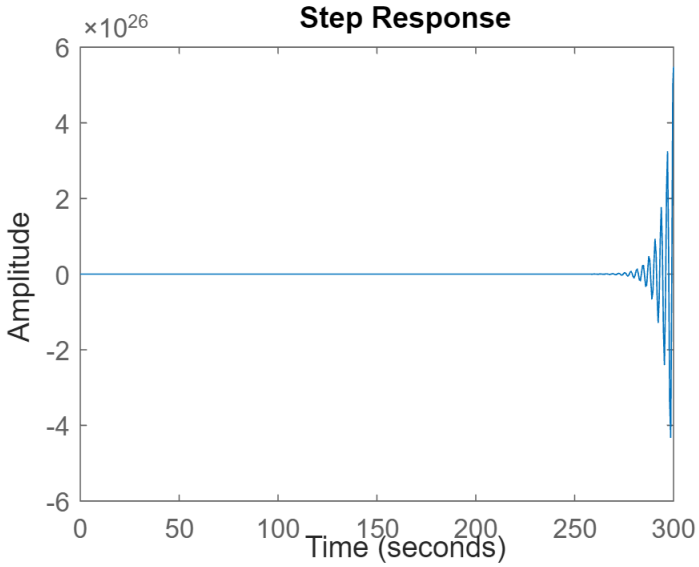
```
% Tu código aquí
s=tf('s');
Gc=(s+1)/(s*(s+5));
G6=20/(s^3+12*s^2+21*s+5);
S1=feedback(series(G_c,G_6),1);
S2=feedback(series(4*G_c,G_6),1);
S3=feedback(series(8.4*G_c,G_6),1);
S4=feedback(series(15*G_c,G_6),1);
step(S1,S2)
```



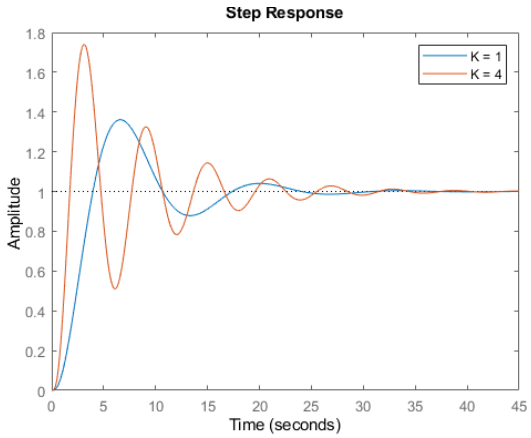

```
step(S3)
```

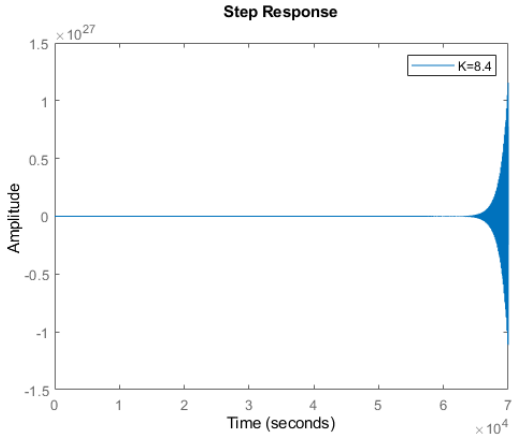


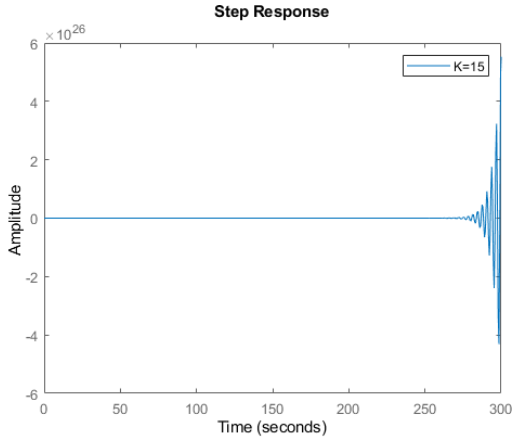

```
step(S4)
```



Resultado esperado:







- *step(sys1, 'f1', sys2, 'f2', …):* Obtiene un resultado análogo al del caso anterior. Las respuestas se representarán con los formatos indicados por 'fi', según los criterios aplicados en la

función plot. Así, por ejemplo, *'g--'* determinaría que la respuesta correspondiente se representaría con una línea discontinua de color verde. Ejemplo:

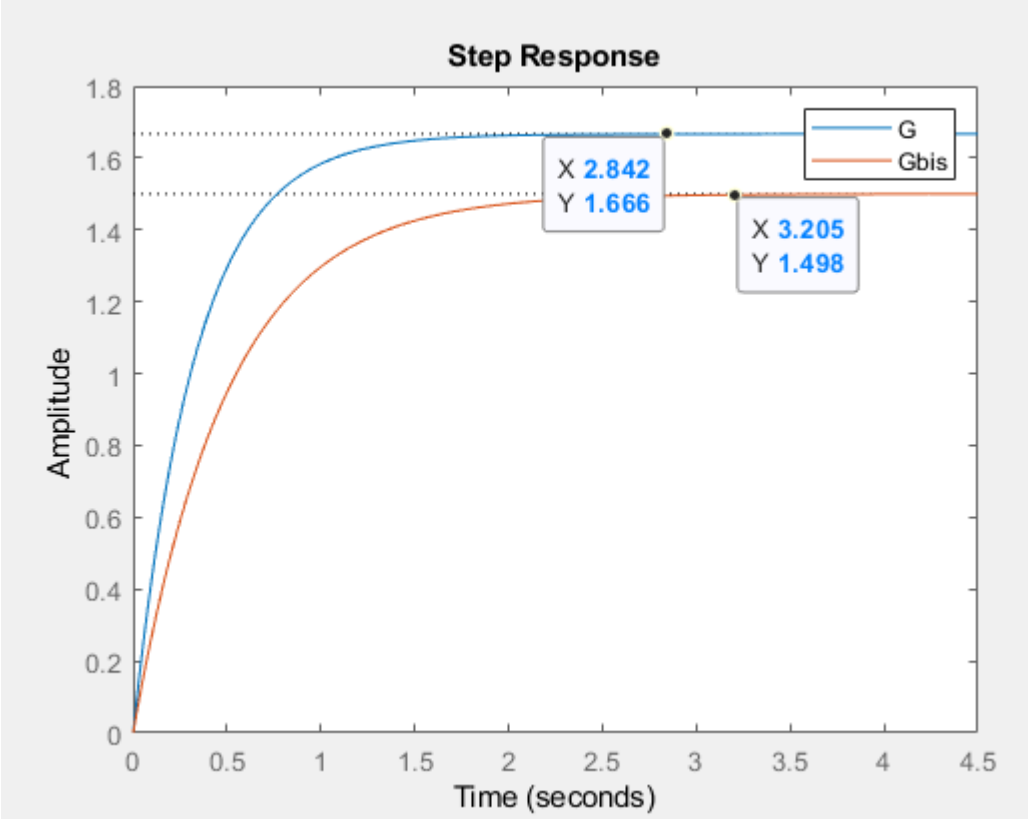
```
step(G,'b.',Gbis,'m--')     % Mostramos la respuesta escalón de ambos con distinto
formato
```

```
Unrecognized function or variable 'Gbis'.
```

```
legend
```

1.2 Jugando con el gráfico

En todos los casos anteriores es posible activar uno o más cursores sobre las diferentes gráficas representadas. Para ello bastaría con pulsar el botón izquierdo del ratón cuando el cursor esté situado sobre la gráfica en cuestión. Junto a cada cursor se indicará: la variable que almacena el modelo del sistema para el que se ha obtenido la respuesta temporal considerada y el instante de tiempo y el valor de la salida asociados a la posición del cursor. Ejemplo:

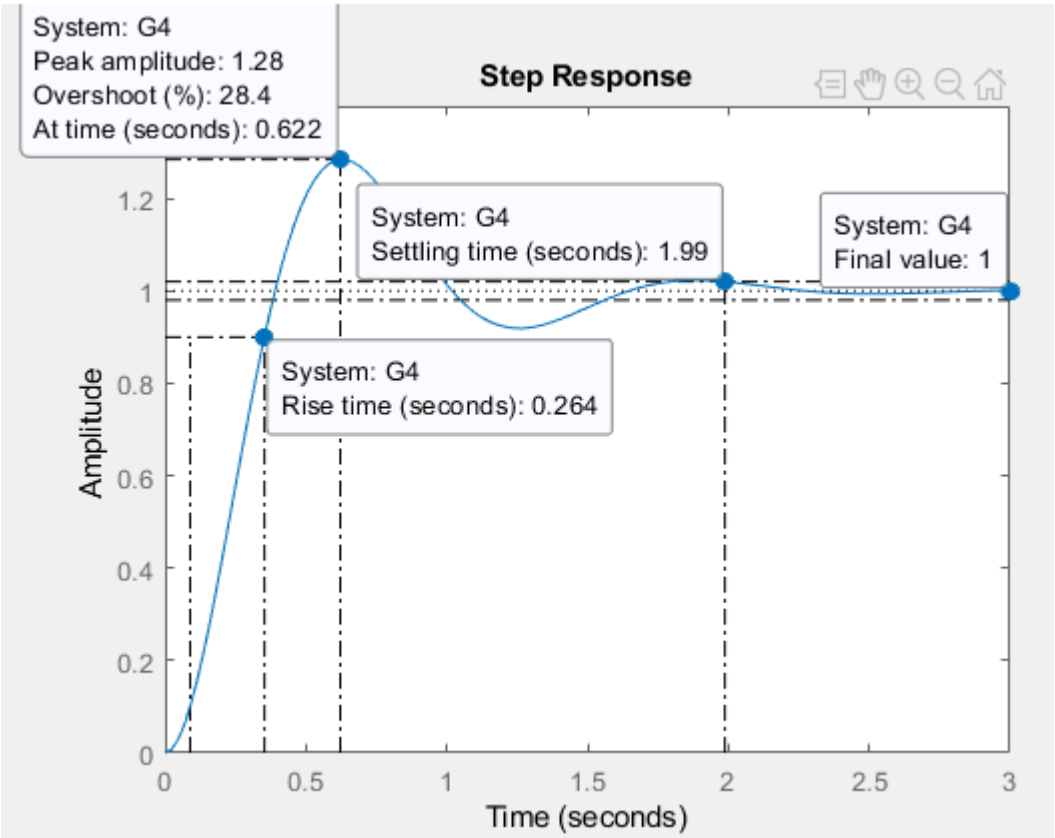


Asimismo, y aunque estudiaremos estos conceptos más adelante, en todos los casos anteriores es posible determinar los valores de ciertas características de las respuestas representadas. En concreto, se puede determinar:

- el pico de la respuesta (máximo sobreimpulso, sobreoscilación y tiempo de pico),

- el tiempo de establecimiento,
- el tiempo de subida, y
- el valor de estabilización.

Para ello bastaría con pulsar el botón derecho del ratón cuando el cursor esté situado en el interior de la correspondiente ventana, seleccionar la opción *Characteristics* en el menú mostrado y elegir la característica a mostrar [peak response (peak amplitude, overshoot y peak time), settling time, rise time y steady state, respectivamente]. Como consecuencia de ello, aparecerá resaltado sobre cada respuesta representada un punto vinculado con dicha característica y pulsando el botón izquierdo del ratón cuando el cursor se encuentre sobre dicho punto se mostrará junto a él el nombre de la variable que almacena el modelo del sistema asociado a la respuesta en cuestión y el nombre y valor de la característica. Ejemplo:



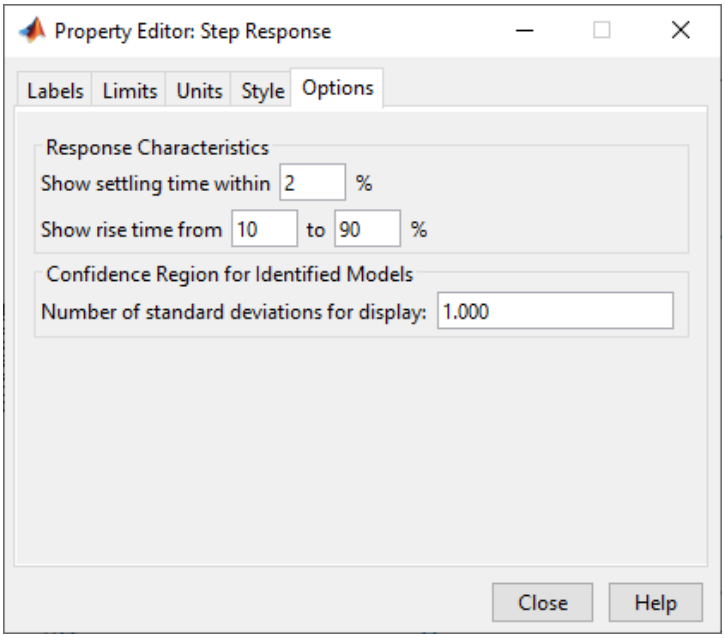
Por otra parte, en todos aquellos casos en los que en una misma gráfica se hayan representado las respuestas de dos o más sistemas; es posible ocultarlas/mostrarlas pulsando el botón derecho del ratón, seleccionando la opción *Systems* en el menú mostrado y seleccionando el nombre de la variable que almacena el modelo del sistema cuya respuesta se desea ocultar/mostrar.

Si se desea, es posible visualizar y ocultar una rejilla en la ventana en la que se realizan las representaciones gráficas de las respuestas. Para ello, bastará con pulsar el botón derecho del ratón y seleccionar la opción *Grid* en el menú mostrado.

Finalmente, es posible modificar algunas propiedades de las gráficas representadas. Entre ellas cabe citar:

- las etiquetas (*labels*) que acompañan a los ejes y a la ventana gráfica,
- los valores límites (*limits*) de la salida y el tiempo considerados en los ejes, y
- determinadas opciones (*options*) asociadas a las características de las respuestas, en concreto, el criterio utilizado para la determinación del tiempo de establecimiento (por defecto es el del 2%) y del tiempo de subida (por defecto es el del 10 al 90%).

Para ello, bastará con pulsar el botón derecho del ratón, seleccionar la opción *Properties*, pinchar, en la ventana de diálogo mostrada a continuación, sobre la pestaña (etiquetada como *labels*, *limits* u *options*) asociada a la propiedad que se desee modificar y realizar el correspondiente cambio. Ejemplo:



1.3 Guardando la salida de la respuesta

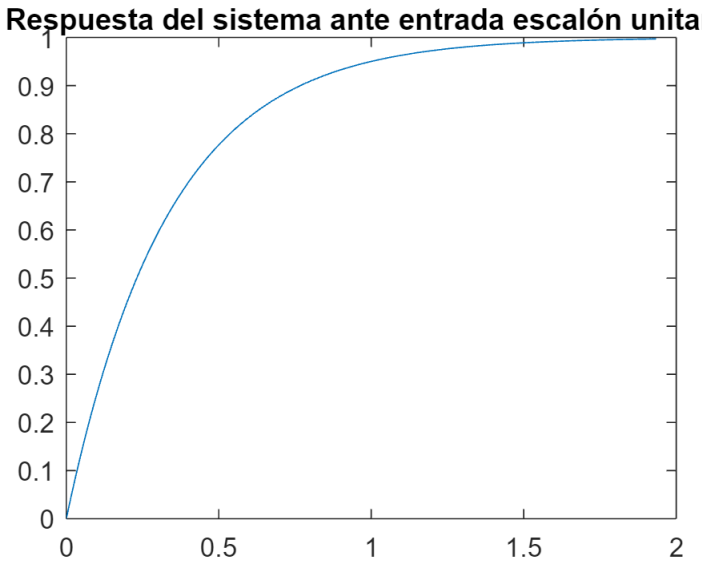
El comando **`step`** puede ejecutarse de tal manera que podamos guardar en variables de MATLAB resultados interesantes sobre su ejecución. Por ejemplo:

- *`[y, t] = step(sys):`* Obtiene dos vectores representativos de la respuesta que presentaría, ante la aplicación de una señal de entrada escalón unitario, un sistema cuyo modelo matemático fuese el almacenado en la variable *`sys`*. En este caso no se obtiene ninguna representación gráfica. *`y`* es el vector o la matriz de los valores de la/s señal/es de respuesta. *`t`* es el vector de tiempos. Para sistemas con una sola entrada, *`y`* tiene igual número de filas que el vector *`t`* y tantas columnas como señales de salida. Para sistemas con múltiples entradas, *`y`* proporciona las respuestas a un escalón para cada canal de entrada, las cuales son diferenciables por medio de una tercera dimensión. Así, y*`(:,:,j)`* proporciona la respuesta a un escalón correspondiente al canal de entrada jésimo.

- *y = step(sys, t):* Obtiene un resultado análogo al del caso anterior. La respuesta se generará para un vector de tiempos dado por el valor de *t*, cuyo formato ha de ser idéntico al descrito más arriba.

El almacenar la respuesta en una variable puede ser intersante para analizarla en detalle, como veremos a lo largo del curso. Además, la respuesta podría visualizarse posteriormente utilizando el comando **plot**. Por ejemplo:

```
G = tf(3,[1 3]);
[y,t] = step(G);
plot(t,y), title('Respuesta del sistema ante entrada escalón unitario')
```

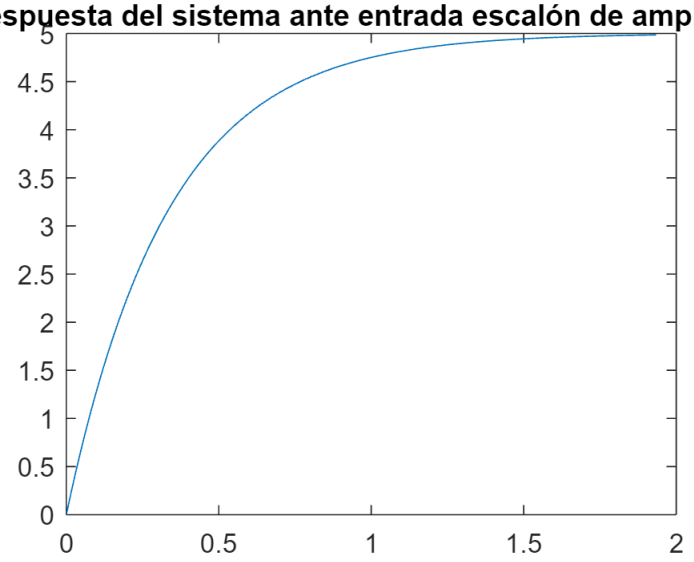


1.4 Escalones de distinta amplitud

Si en vez de simular la respuesta del sistema ante entrada escalón unitario, queremos simularla ante un escalón de una amplitud arbitraria, bastaría con multiplicar el modelo del sistema por dicha amplitud. Por ejemplo, la siguiente celda de código realiza una simulación ante escalón de amplitud 5:

```
G = tf(3,[1 3]);
[y,t] = step(5*G);
plot(t,y), title('Respuesta del sistema ante entrada escalón de amplitud 5')
```

Respuesta del sistema ante entrada escalón de amplitud

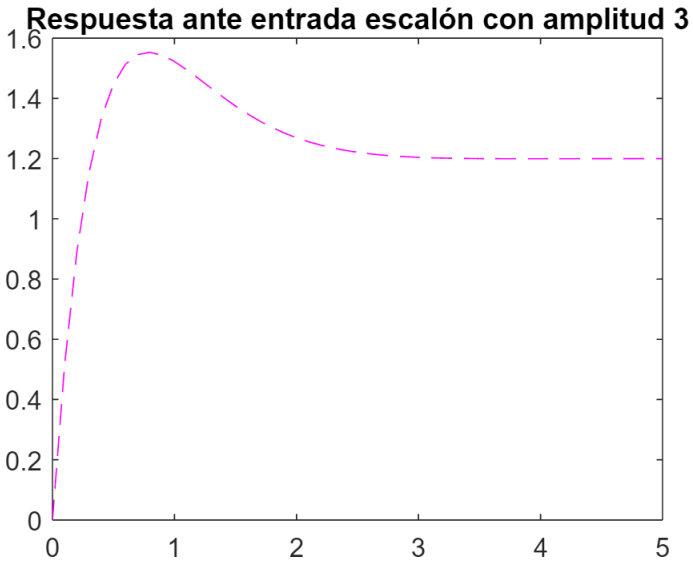


Tarea 5: Se pide obtener la respuesta ante entrada escalón de amplitud 3 desde $t = 0s$ hasta $t = 5s$ de la planta definida por la función de transferencia $G(s) = \frac{2(s+1)}{(s+2+1i)(s+2-1i)}$, y graficarla en color magenta y utilizando un estilo de línea de guiones.

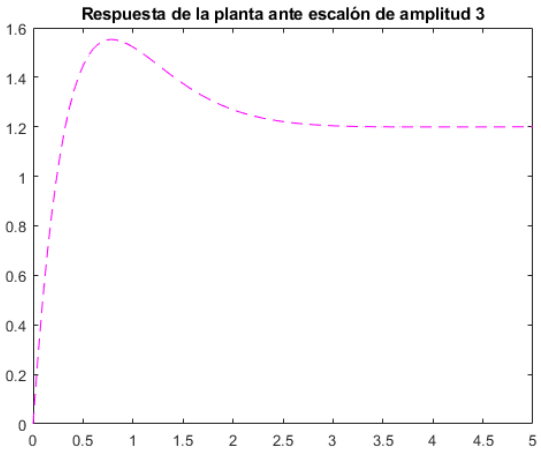
```
% Tu código aquí
G=zpk(-1,[-2+1*i -2-1*i],2)
```

```
G =

     2 (s+1)
  --------------
  (s^2 + 4s + 5)

Continuous-time zero/pole/gain model.
Model Properties
```

```
[y,t]= step(3*G,0:0.1:5);
plot(t,y,'m--'),title('Respuesta ante entrada escalón con amplitud 3')
```



Resultado esperado:



2. El impulso unitario

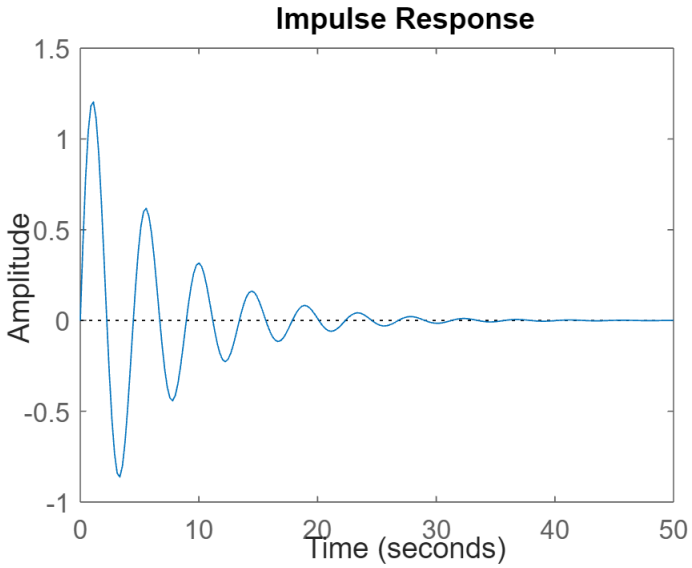
Por otra parte, en caso de sustituir, en los formatos anteriormente presentados de llamada a la función **`step`**, el nombre de ésta por **`impulse`**, se obtendrían los resultados análogos correspondientes a la consideración de una señal de entrada impulso unitario. De este modo:

- *`impulse( sys , t )`*: simula la respuesta ante entrada impulso del sistema modelado en `sys` (ya sea en descripción externa o interna) hasta el instante de tiempo `t` (opcional).

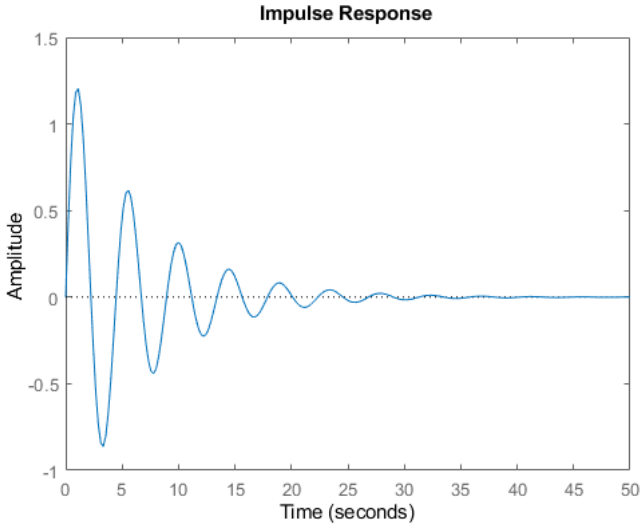
Las mismas posibilidades vistas para el comando **`step`** en cuanto a simulación de diversos sistenas, guardado de la salida de la respuesta, y entradas de distinta amplitud, también están disponibles para **`impulse`**.

Tarea 6: Esta tarea propone la obtención de la respuesta ante impulso unitario del sistema $G(s) = \frac{2}{s^2 + 0.3s + 2}$ hasta el instante de tiempo $t = 50s$.

```
% Tu código aquí
G=tf(2,[1 0.3 2]);
impulse(G,50)
```



Resultado esperado:



3. Rampa/parábola unitaria

En este caso, MATLAB no nos provee de un comando concreto para la simulación de la respuesta de sistemas ante entrada rampa o parábola unitarias. Pero no está todo perdido, ¡podemos emplear el comando **`step`** para este fin!

Poniendo el caso de la función de transferencia $G(s)$ de un sistema, recordamos que la respuesta de este se define como $C(s) = G(s)U(s)$. Sabemos que:

- Entrada escalón unitaria: $u(t) = 1 \rightarrow U_e(s) = \frac{1}{s}$.
- Entrada rampa unitaria: $u(t) = t \rightarrow U_r(s) = \frac{1}{s^2}$.
- Entrada parábola unitaria $u(t) = t^2 \rightarrow U_p(s) = \frac{1}{2s^3}$

Por ejemplo, para el caso de la entrada rampa unitaria:

$$C(s) = G(s)U_r(s) = G(s)\frac{1}{s^2} = \frac{G(s)}{s}\frac{1}{s} = \frac{G(s)}{s}U_e(s)$$

Por lo que podremos obtener la respuesta rampa unitaria empleando el comando **`step`** previa multiplicación de la función de transferencai $G(s)$ por $1/s$:

```
s = tf('s');
step(G(s)*1/s)
```

Para el caso de la parábola unitaria, resultaría:

```
s = tf('s');
step(G(s)*1/(2*s^2))
```

Tarea 7: Se pide simular la respuesta del sistema $G(s) = \frac{1}{s^2 + 2s + 2}$ ante entrada rampa unitaria hasta el instante $t = 10s$, graficando en la misma ventana tanto la entrada en color azul como la respuesta del sistema en color rojo y punteada. Muestra en una leyenda con qué información se corresponde cada línea.

*Pista: **`help legend`** puede serte de ayuda.*

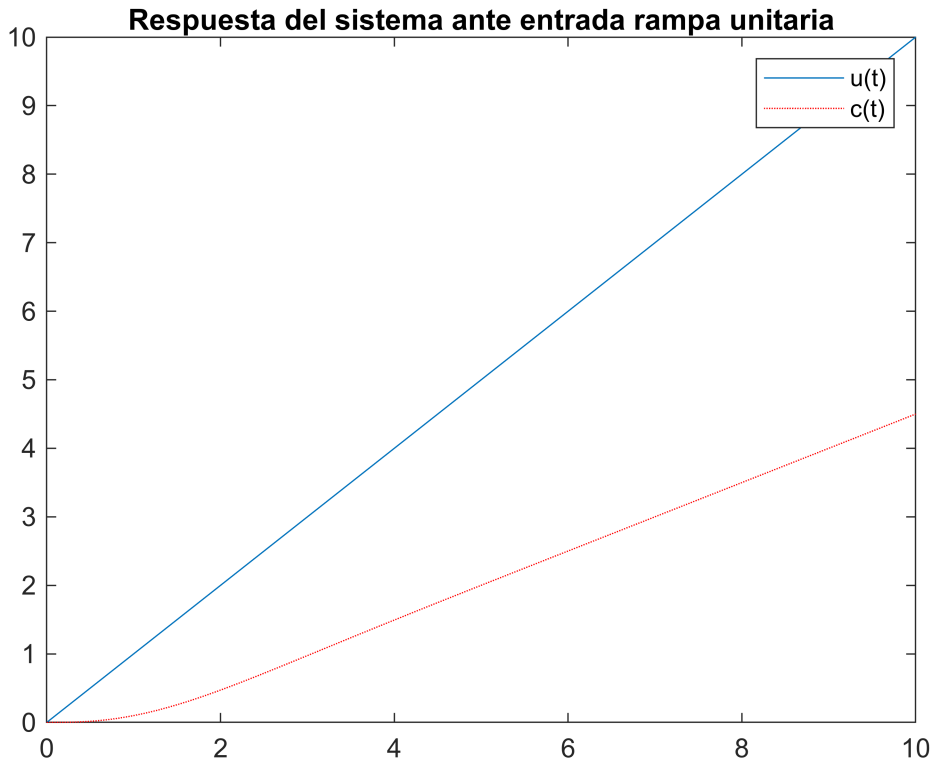
```
% Tu código aquí
help legend
```

```
legend - Add legend to axes
   This MATLAB function creates a legend with descriptive labels for each
   plotted data series.

   Syntax
     legend
     legend(label1,...,labelN)
     legend(labels)
     legend(subset,___)
     legend(target,___)

     legend(___,'Location',lcn)
```

```
  legend(___,'Orientation',ornt)
  legend(___,Name,Value)
  legend(bkgd)
  lgd = legend(___)

  legend(vsbl)
  legend('off')

Input Arguments
  label1,...,labelN - Labels (as separate arguments)
    character vectors | strings
  labels - Labels (as an array)
    cell array of character vectors | string array | categorical array
  subset - Data series to include in legend
    vector of graphics objects
  target - Target for legend
    Axes object | PolarAxes object | GeographicAxes object |
    standalone visualization
  lcn - Legend location
    'north' | 'south' | 'east' | 'west' | 'northeast'
  ornt - Orientation
    'vertical' (default) | 'horizontal'
  bkgd - Legend box display
    'boxon' (default) | 'boxoff'
  vsbl - Legend visibility
    'hide' | 'show' | 'toggle'

Name-Value Arguments
  TextColor - Text color
    [0 0 0] (default) | RGB triplet | hexadecimal color code | 'r' |
    'g' | 'b'
  FontSize - Font size
    scalar value greater than zero
  NumColumns - Number of columns
    1 (default) | positive integer

Output Arguments
  lgd - Legend object
    Legend object

Examples
  Add Legend to Current Axes
  Add Legend to Specific Axes
  Specify Legend Labels During Plotting Commands
  Exclude Line from Legend
  List Entries in Columns and Specify Legend Location
  Reverse Order of Legend Items
  Display Shared Legend in Tiled Chart Layout
  Included Subset of Graphics Objects in Legend
  Create Legend with LaTeX Markup
  Add Title to Legend
  Remove Legend Background
  Specify Legend Font Size and Color

See also plot, hold, title, xlabel, ylabel, text, Legend

Introduced in MATLAB before R2006a
Documentation for legend
```

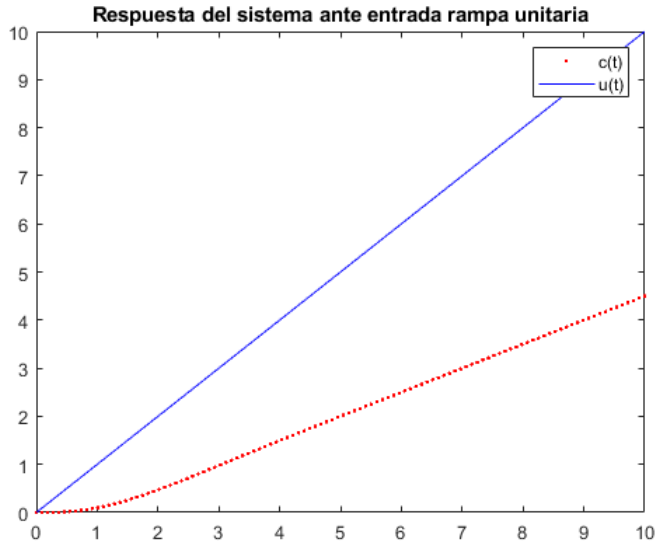
```
s = tf('s');
G=tf(1,[1 2 2]);
t=0:0.1:10;
y=step(G*(1/s),t);
```

```
plot(t,t,t,y,'r:'),title('Respuesta del sistema ante entrada rampa unitaria');
legend ('u(t)','c(t)')
```




```

```

Resultado esperado:



4. Creando una señal de entrada arbitraria

Si queremos analizar la respuesta temporal de nuestro sistema ante una entrada arbitraria distinta de las anteriores, siempre podremos recurrir al comando **`lsim`**. Su sintaxis es:

[δy,t,δx] = lsim(G,δu,t,δx0)

donde:

- *δy*: respuesta temporal del sistema linealizado para cada uno de los valores del vector *t*.
- *t*: vector de tiempos utilizado para la simulación
- *δx*: variables de estado linealizadas calculadas para cada uno de los valores del vector *t* (usado con descripción interna).
- *G*: sistema dinámico a simular. Para el caso de una descripción interna viene dado por *G=ss(A,B,C,D)*.
- *δu*: entrada o actuación, con el mismo número de elementos que el vector *t*.
- *δx0*: vector de condiciones iniciales.

Al igual que con el comando step, se puede simular la respuesta de más de un sistema, incluyendo sus modelos a continuación de *G*, incluso especificar su formato. De igual modo, si se llama sin argumentos de salida, se mostrará en una ventana gráfica la respuesta del sistema.

Ejemplo: La siguiente celda de código muestra gráficamente la respuesta de un sistema ante una entrada escalón de amplitud 3 que se inicia en el instante $t = 0s$ y termina en $t = 2s$.

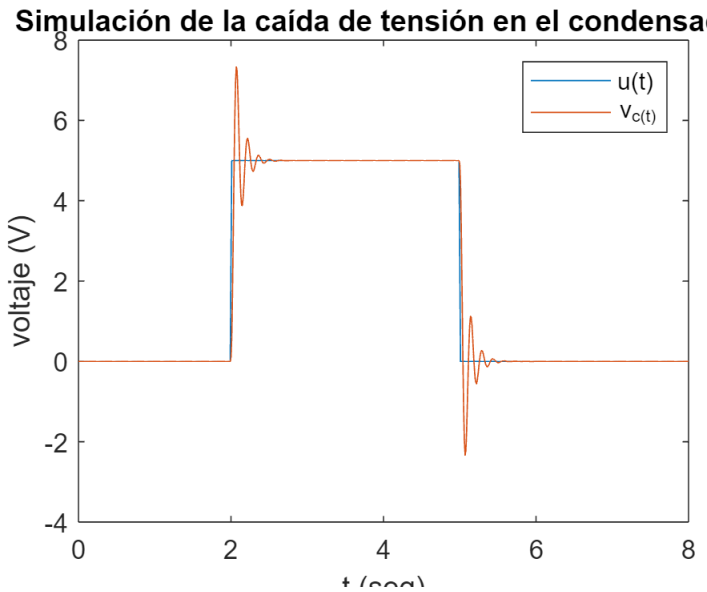
```
inc = 0.01;
t = 0:inc:10;
u = zeros(length(t),1);
u(1:2/inc+1) = 3; % Matlab empieza indexando en 1
G = tf(2,[1 2 1]);
y = lsim(G,u,t);
plot(t,u,t,y); legend('u(t)','c(t)')
```

Tarea 8: Se considera la siguiente función de transferencia que modela un circuito LRC, siendo la entrada el voltaje con el que se alimenta al circuito y la salida la caída de voltaje en el condensador:

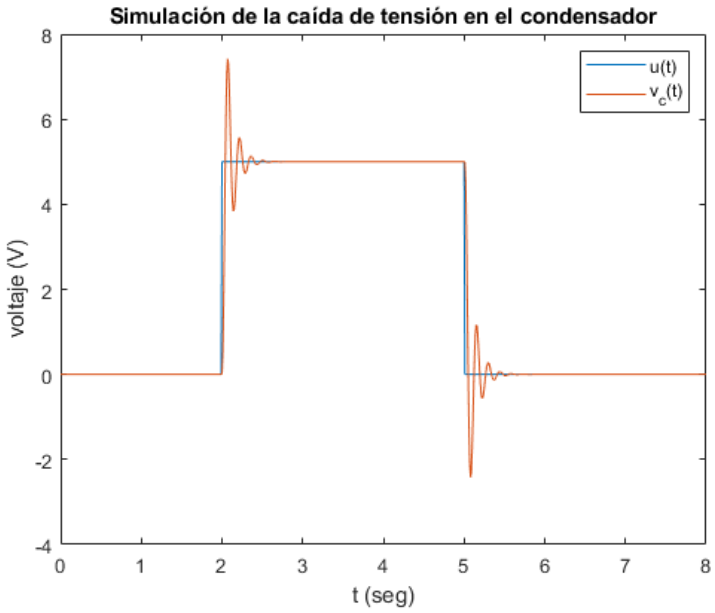
$$G_{\text{circuito}}(s) = \frac{2000}{s^2 + 20s + 2000}$$

Se pide graficar la respuesta del sistema durante 8 segundos ante una entrada escalón de amplitud 5 voltios, recreando un escenario donde se alimenta al circuito desde el instante $t = 2s$ hasta $t = 4s$.

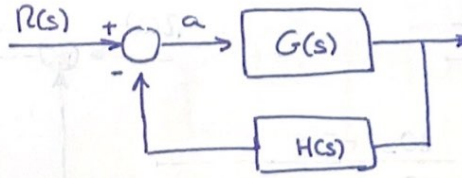
```
% Tu código aquí
G=tf(2000,[1 20 2000]);
t=0:0.01:8;
u=5*(heaviside(t-2)-heaviside(t-5));
y=lsim(G,u,t);
plot(t,u,t,y),title('Simulación de la caída de tensión en el condensador')
legend('u(t)','v_c(t)')
xlabel('t (seg)');
ylabel('voltaje (V)');
```



Resultado esperado:



Tarea 4



$$q = \begin{bmatrix} 2 & +1 & -3 \\ 3 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$a = R(s) - Y \cdot H$

$Y = a \cdot G = R(s) \cdot G(s) - G \cdot Y \cdot H.$

$Y[1 + G \cdot H] = R \cdot G \rightsquigarrow Y(s) = \dfrac{R(s)\,G(s)}{1 + G(s) \cdot H(s)}$

Función de Transf $\rightsquigarrow$ $\boxed{\dfrac{Y(s)}{R(s)} = \dfrac{G(s)}{1 + G(s) \cdot H(s)}}$

$$F_1 = \frac{\frac{s^2+5s+2}{s^3+3s^2+3s+1}}{1 + \frac{s^2+5s+2}{s^3+3s^2+3s+1} \cdot 1} = \frac{(s^3+3s^2+3s+1) \cdot s^2+5s+2}{(s^3+3s^2+3s+1) \cdot [(s^3+3s^2+3s+1) + s^2+5s+2]}$$

$$= \frac{s^2+5s+2}{s^3+4s^2+8s+3.}$$

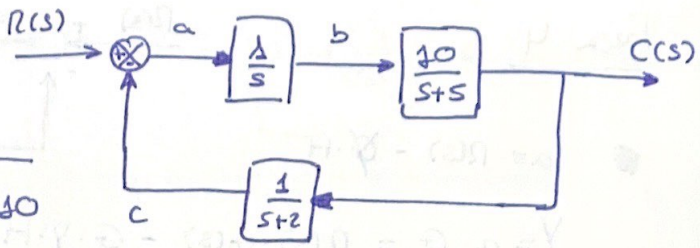
$$F_2 = \frac{\frac{7s^2+s+2.}{s^3+s^2+1.}}{1 + \frac{7s^2+s+2}{s^3+s^2+1} \cdot \frac{1}{(s+1)}} = \frac{(7s^2+s+2)(s^3+s^2+1) \cdot (s+1)}{[(s^3+s^2+1)(s+1) + (7s^2+s+2) \cdot 1.] (s^3+s^2+1)}$$

$$= \frac{(7s^2+s+2)(s+1)}{[(s^3+s^2+1)(s+1) + (7s^2+s+2)]} \cdot = \frac{7s^3+8s^2+3s+2}{s^4+2s^3+8s^2+2s+3.}$$

$$F_3 = \frac{\frac{s+1}{s^2+5s}}{1 + \frac{s+1}{s^2+5s} \cdot \frac{s+2}{s^2+4s+5}} = \frac{(s+1) \cdot (s^2+5s) \cdot (s^2+4s+5)}{[(s^2+5s) \cdot (s^2+4s+5) + (s+1)(s+2)] \cdot (s^2+5s)}$$

$$= \frac{(s+1)(s^2+4s+5)}{[(s^2+5s) \cdot (s^2+4s+5) + (s+1)(s+2)]} \cdot = \frac{s^3+5s^2+9s+5.}{s^4+9s^3+26s^2+28s+2}$$

Tarea 5



$$\frac{C(s)}{R(s)} = \frac{10s + 20}{s^3 + 7s^2 + 10s + 10}$$

$a = R(s) + c = R(s) + C(s) \cdot \frac{1}{s+2}$

$b = \frac{1}{s} \cdot a.$

$C(s) = b \cdot \frac{10}{s+5}$

$b = \frac{1}{s} R(s) + C(s) \cdot \frac{1}{s} \cdot \frac{1}{s+2}$

$c = C(s) \cdot \frac{1}{s+2}$

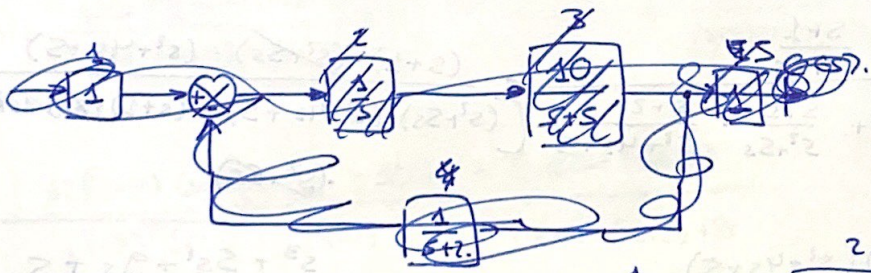
$C(s) = \frac{1}{s} \cdot \frac{10}{s+5} \cdot R(s) + C(s) \frac{1}{s} \cdot \frac{1}{s+2} \cdot \frac{10}{s+5}.$

$$C(s) = \frac{10}{s^2+5s} R(s) + C(s) \cdot \frac{10}{s^3+7s^2+10s}.$$

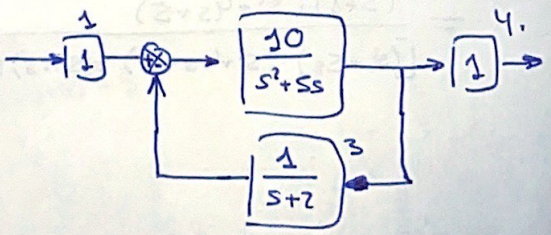
$$C(s) - C(s) \cdot \frac{10}{s^3+7s^2+10s} = \frac{10}{s^2+5s} R(s).$$

$$G(s) = \frac{C(s)}{R(s)} = \frac{1/(s^2+5s)}{1 - \frac{10}{s^3+7s^2+10s}}$$

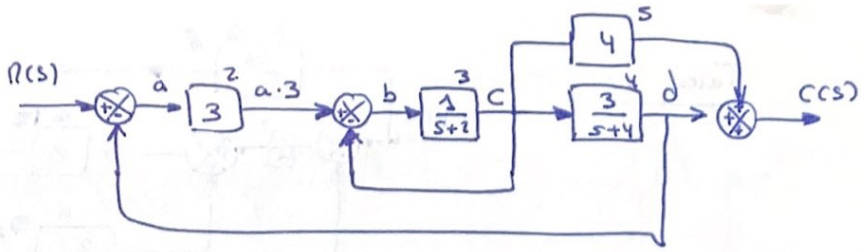
$$G(s) = \frac{s^3+7s^2+10s}{\left[s^3+7s^2+10s-10\right](s^2+5s)} = \frac{10s+5}{s^3+7s^2+10s-10}$$



$$q = \begin{bmatrix} 2 & 1 & -13 \\ 3 & 2 & 0 \\ 4 & 2 & 0 \end{bmatrix}$$



Tarea 6:



$C(s) = 4c + d$

$a = R(s) - d.$

$b = a \cdot 3 - c. = 3R(s) - 3d - c.$

$c = b \cdot \frac{1}{s+2} = \frac{3R(s)}{s+2} - \frac{3d}{s+2} - \frac{c}{s+2.} \Rightarrow \boxed{c = \frac{3R(s) - 3d.}{\left(1 + \frac{1}{s+2.}\right) s+2} = \frac{3R(s) - 3d.}{s+2+1} = \frac{3R(s) - 3d}{s+3.}}$

$d = c \cdot \frac{3}{s+4}$

$d = \frac{9R(s) - 9d}{(s+4)(s+3)} = \frac{9R(s)}{(s+4)(s+3)} - \frac{9d}{(s+4)(s+3)}$

$$\boxed{d = \frac{\frac{9R(s)}{(s+4)(s+3)}}{1 + \frac{9}{(s+4)(s+3)}} = \frac{9R(s)}{(s+4)(s+3) + 9}}$$

$$C(s) = \frac{12R(s) - 12d}{s+3} + \frac{9(R(s))}{(s+4)(s+3)+9.}$$

~~$C(s) = \frac{12R(s) - 12[(s+4)(s+3)+9] \cdot 12c}{s+3}$~~

$$C(s) = \frac{12R(s) - 12\,\frac{9R(s)}{(s+4)(s+3)+9}}{s+3} + \frac{9R(s)}{(s+4)(s+3)+9}$$

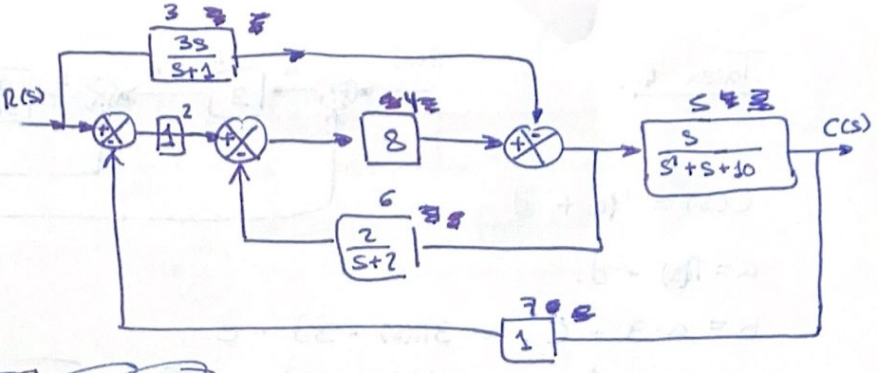
$$C(s) = \frac{12R(s)[(s+4)(s+3)+9] - 108R(s)}{(s+3)^2(s+4) + 9(s+3)} + \frac{9R(s)}{(s+4)(s+3)+9}$$

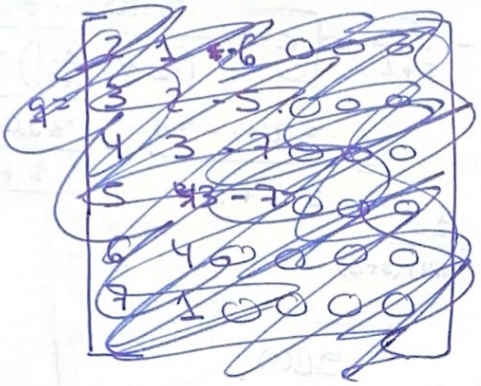
$$\frac{C(s)}{R(s)} = \frac{12[(s+4)(s+3)+9] - 108}{(s+3)^2(s+4) + 9(s+3)} + \frac{9}{(s+4)(s+3)+9}$$

$$\frac{C(s)}{R(s)} = \frac{12(s+4)(s+3)}{(s+3)^2(s+4) + 9(s+3)} + \frac{9}{(s+4)(s+3)+9}$$

$$\boxed{\frac{C(s)}{R(s)} = \frac{12s + 57}{(s+3)(s+4)+9}}$$

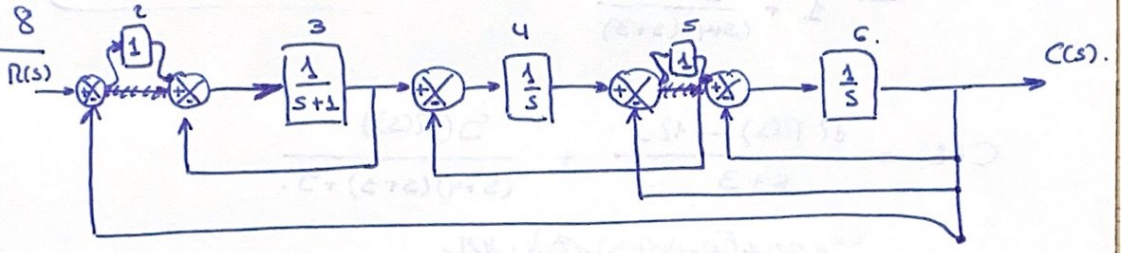
Tarea 7:





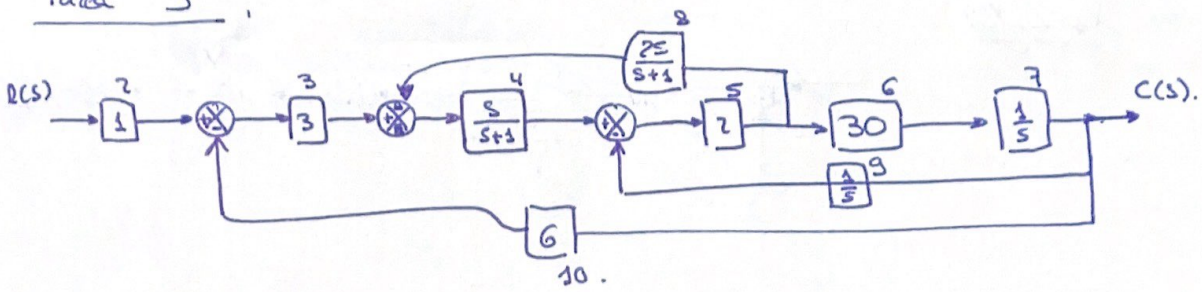
$$q = \begin{bmatrix} 2 & 1 & -7. \\ 3 & 1. & 0 \\ 4 & 2 & -6. \\ 5 & 4 & -3. \\ 6 & 4 & -3. \\ 7 & 5 & 0 \end{bmatrix}$$

Tarea 8



$$q = \begin{bmatrix} 2 & 1 & -6. \\ 3 & 2 & -3 \\ 4 & 3 & -5 \\ 5 & 4 & -6. \\ 6 & 5 & -6. \\ 7 & 6. & 0. \end{bmatrix}$$

Tarea 9 :



$$q = \begin{bmatrix} 2 & 1 & 0 \\ 3 & 2 & -10 \\ 4 & 3 & -8 \\ 5 & 4 & -9 \\ 6 & 5 & 0 \\ 7 & 6 & 0 \\ 8 & 5 & 0 \\ 9 & 7 & 0 \\ 10 & 7 & 0 \\ 11 & 7 & 0 \end{bmatrix}$$