Guión de la Práctica 3 Fundamentos de la Automática:

1.Polos y ceros de una función de transferencia de MATLAB

Ejercicio práctico 1. Extracción y dibujo de polos y ceros de funciones de transferencia

1. Extrae los valores numéricos de los polos y ceros de las siguientes funciones de transferencia:

```
G1 = tf(1,[1 0 4])
```

```
G1 =

     1
  -------
  s^2 + 4

Continuous-time transfer function.
```

```
polos1 = pole(G1)
```

```
polos1 = 2×1 complex
   0.0000 + 2.0000i
   0.0000 - 2.0000i
```

```
ceros1 = zero(G1)
```

```
ceros1 =

  0×1 empty double column vector
```

```
[polos1,ceros1] = pzmap(G1)
```

```
polos1 = 2×1 complex
   0.0000 + 2.0000i
   0.0000 - 2.0000i
ceros1 =

  0×1 empty double column vector
```

```
G2 = tf(1,[1 4 6 4 1])
```

```
G2 =

                1
  ----------------------------
  s^4 + 4 s^3 + 6 s^2 + 4 s + 1

Continuous-time transfer function.
```

```
polos2 = pole(G2)
```

```
polos2 = 4×1 complex
  -1.0002 + 0.0000i
  -1.0000 + 0.0002i
  -1.0000 - 0.0002i
  -0.9998 + 0.0000i
```

```
ceros2 = zero(G2)
```

```
ceros2 =

  0×1 empty double column vector
```

```
[polos2,ceros2] = pzmap(G2)
```

```
polos2 = 4×1 complex
  -1.0002 + 0.0000i
  -1.0000 + 0.0002i
  -1.0000 - 0.0002i
  -0.9998 + 0.0000i
ceros2 =

  0×1 empty double column vector
```

```
G3 = tf([1 2 1],[1 0 -1])
```

```
G3 =

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

Continuous-time transfer function.
```

```
polos3 = pole(G3)
```

```
polos3 = 2×1
    -1
     1
```

```
ceros3 = zero(G3)
```

```
ceros3 = 2×1
    -1
    -1
```

```
[polos3,ceros3] = pzmap(G3)
```

```
polos3 = 2×1
    -1
     1
ceros3 = 2×1
    -1
    -1
```

```
G4 = tf([1 0 -1],[1 5 11 25 34 20 24])
```

```
G4 =

                       s^2 - 1
  -------------------------------------------------
  s^6 + 5 s^5 + 11 s^4 + 25 s^3 + 34 s^2 + 20 s + 24

Continuous-time transfer function.
```

```
polos4 = pole(G4)
```

```
polos4 = 6×1 complex
  -3.0000 + 0.0000i
  -2.0000 + 0.0000i
```

```
  -0.0000 + 2.0000i
  -0.0000 - 2.0000i
  -0.0000 + 1.0000i
  -0.0000 - 1.0000i
```

```
ceros4 = zero(G4)
```

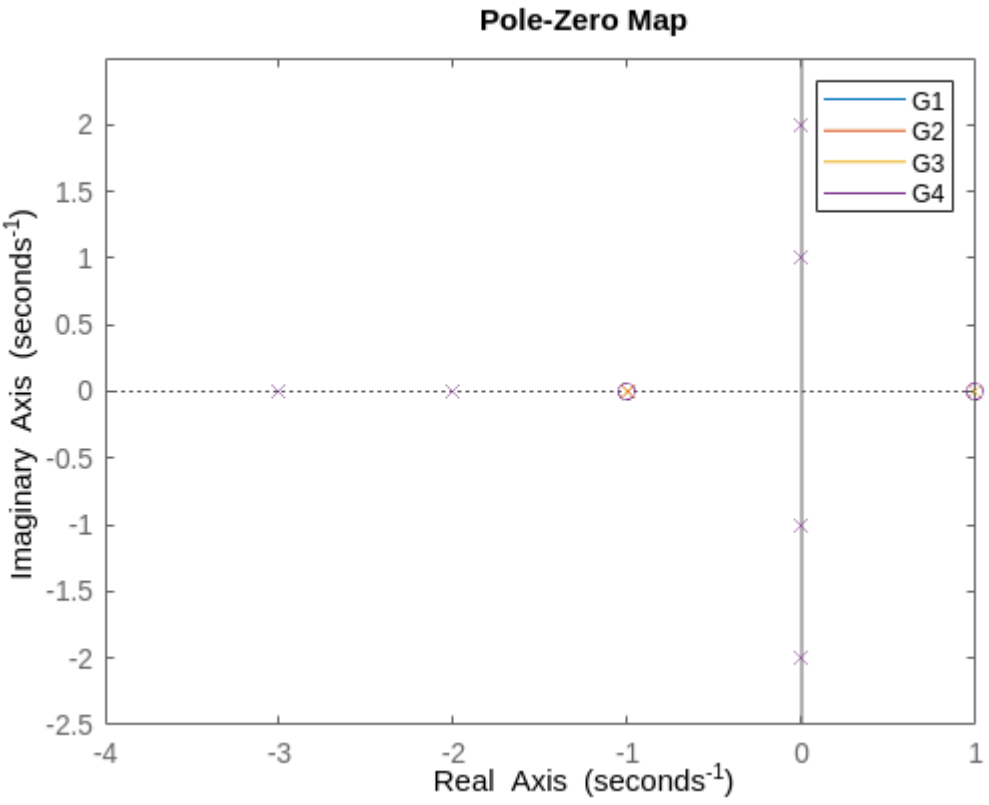
```
ceros4 = 2×1
    -1
     1
```

```
[polos4,ceros4] = pzmap(G4)
```

```
polos4 = 6×1 complex
  -3.0000 + 0.0000i
  -2.0000 + 0.0000i
  -0.0000 + 2.0000i
  -0.0000 - 2.0000i
  -0.0000 + 1.0000i
  -0.0000 - 1.0000i
ceros4 = 2×1
    -1
     1
```

2. Para cada una de las funciones de transferencia, realiza una gráfica del plano complejo donde aparezcan dichos polos y ceros.

```
pzmap(G1,G2,G3,G4)
legend('G1','G2','G3','G4')
```



Ejercicio práctico 2. Polos y ceros en lazo abierto y en lazo cerrado en función de K

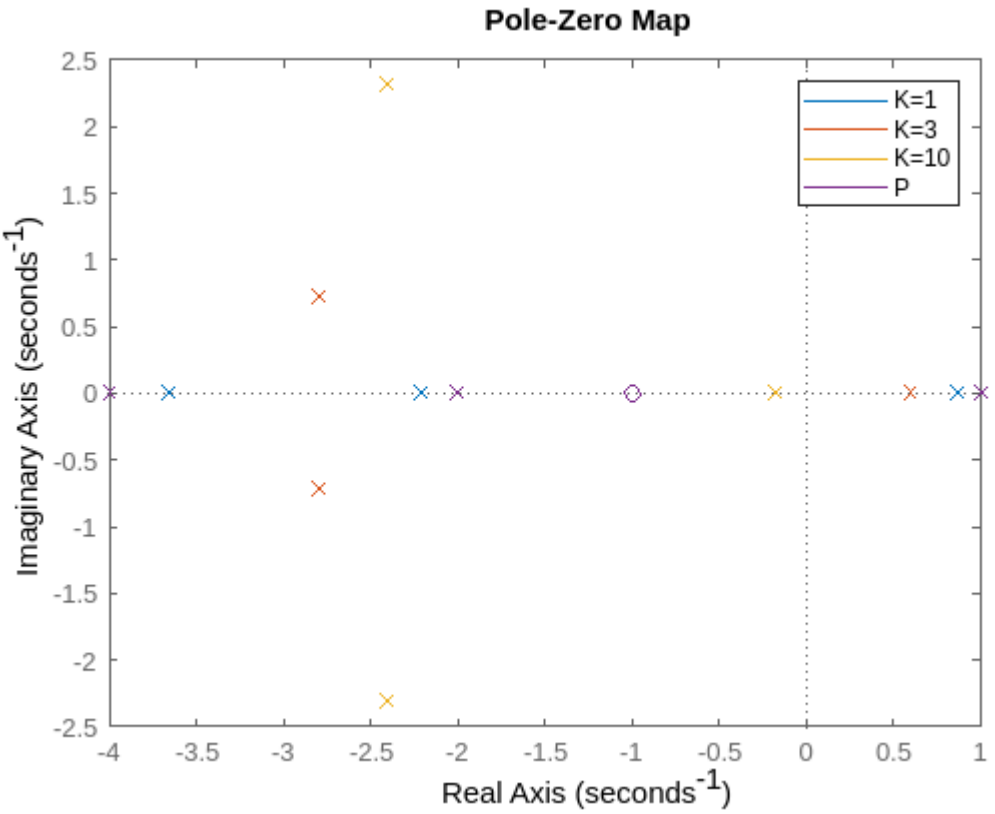
1. Realiza una gráfica del plano complejo con los polos y ceros de P(s):

P(s) = s+1/(s-1)(s+2)(s+4)

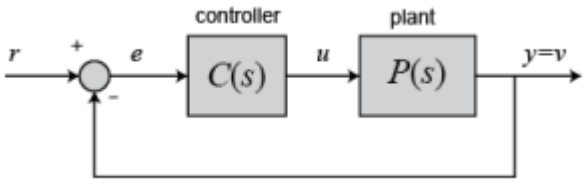
```
P = zpk(-1,[1 -2 -4],1)
```

```
P =

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

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

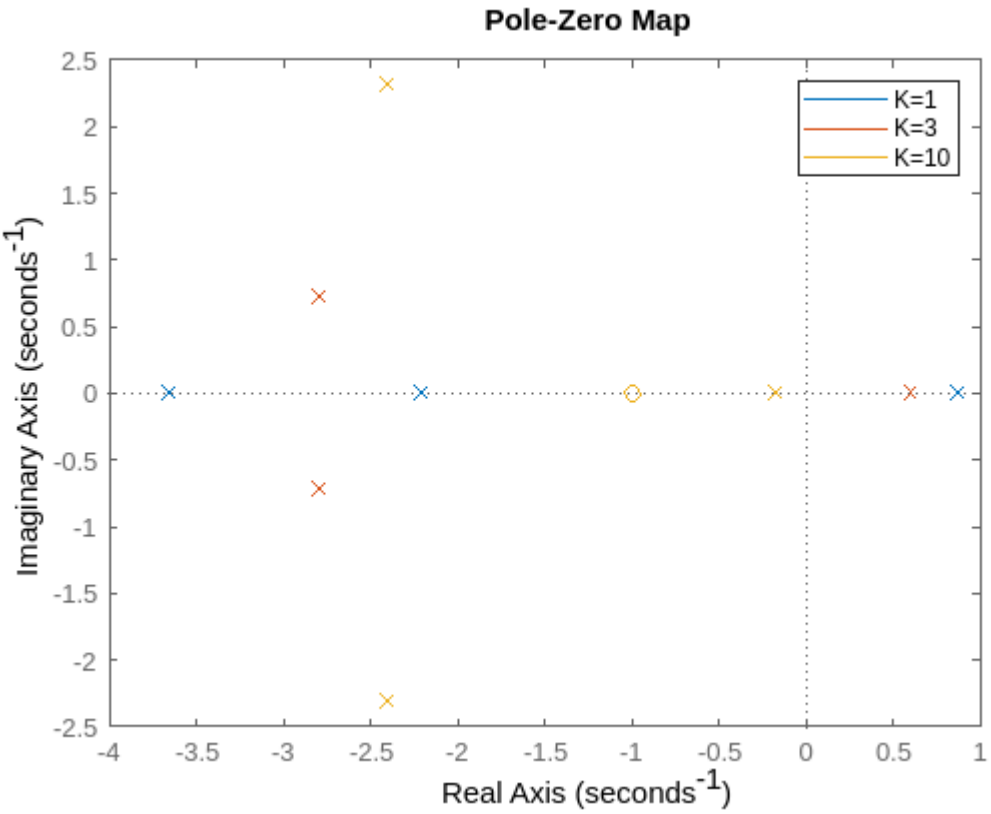
```
pzmap(P)
```



2. Realiza ahora una única gráfica de los polos y ceros de la función de transferencia en lazo cerrado GLC(s) del sistema de la figura, con la misma P(s) y con C(s) = K, para tres valores de la constante K, K=1, K=3 y K=10.



```
K = [1 3 10];
figure; hold on
for t=1:length(K)
    pzmap(feedback(K(t)*P,1));
end
legend('K=1','K=3','K=10');
```



¿Influye el valor de K en la posición de los polos, de los ceros o de ambos en el sistema en lazo cerrado? Justifica tu respuesta.

2. Estudio de la respuesta transitoria de sistemas de control

2.1 Respuesta temporal a entradas impulso y escalón unitarios

Ejercicio práctico 3. Respuesta temporal de sistemas de primer orden

1. Construye cinco funciones de transferencia de primer orden situando sus polos en s = -0.5, s=-1, s = -3, s = -5 y s = -10 respectivamente.

```
G1 = zpk([],-0.5,1)
```

```
G1 =

     1
  -------
```

```
  (s+0.5)

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

```
G2 = zpk([],-1,1)
```

```
G2 =

    1
  -----
  (s+1)

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

```
G3 = zpk([],-3,1)
```

```
G3 =

    1
  -----
  (s+3)

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

```
G4 = zpk([],-5,1)
```

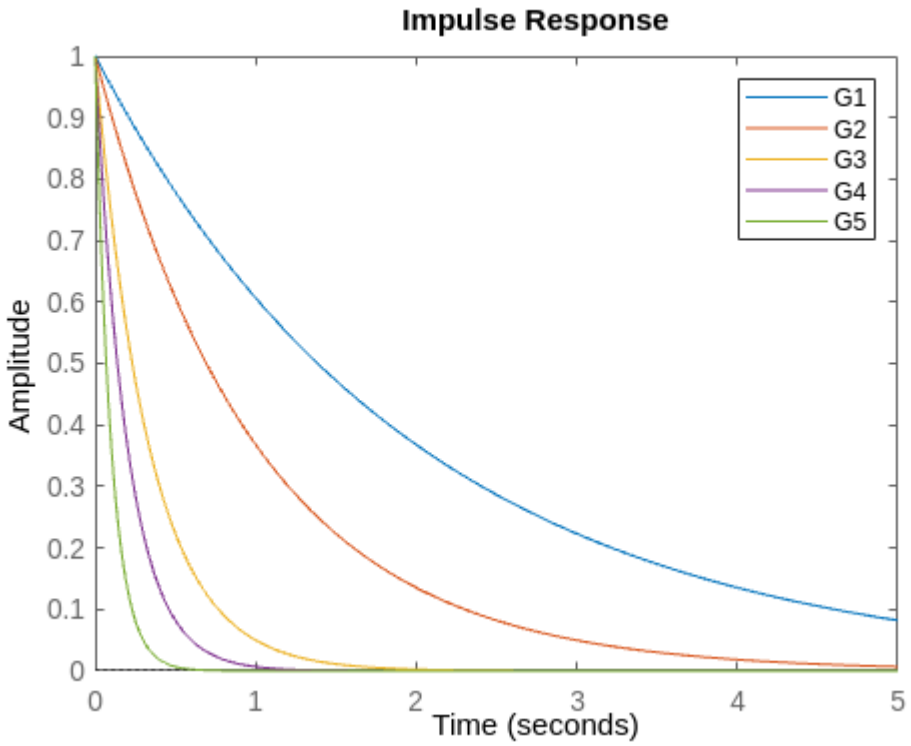
```
G4 =

    1
  -----
  (s+5)

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

```
G5 = zpk([],-10,1)
```

```
G5 =

    1
  ------
  (s+10)

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

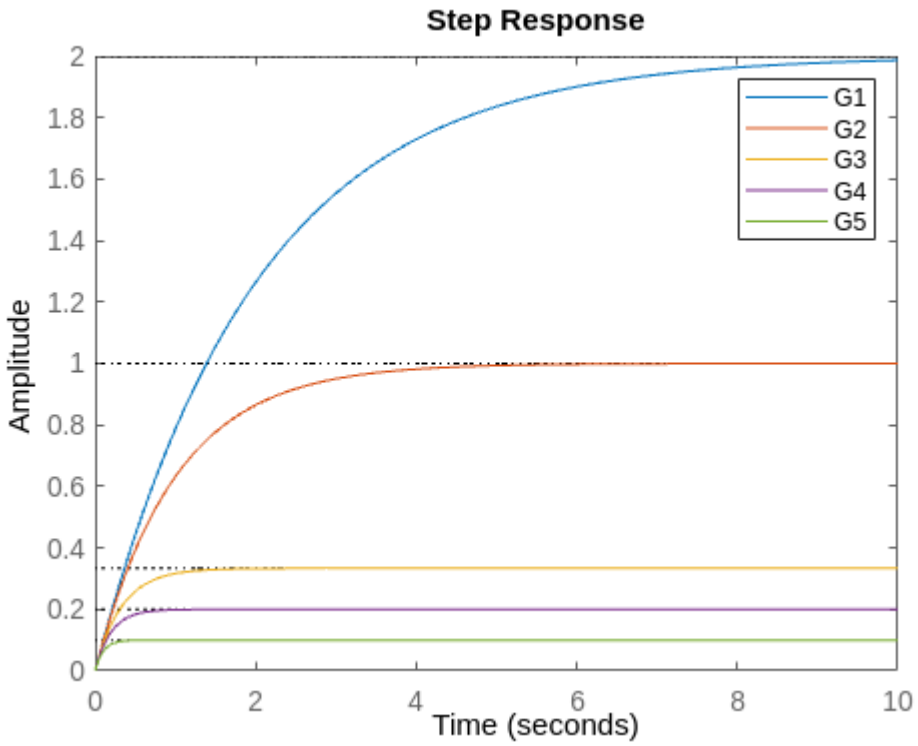
2. Realiza una gráfica que incluya la respuesta al impulso unitario de todas ellas hasta un valor final de t = 5 s. Incluye la leyenda. Justifica el resultado obtenido.

```
figure; hold on
impulse(G1,G2,G3,G4,G5,5)
legend('G1','G2','G3','G4','G5'); hold off
```



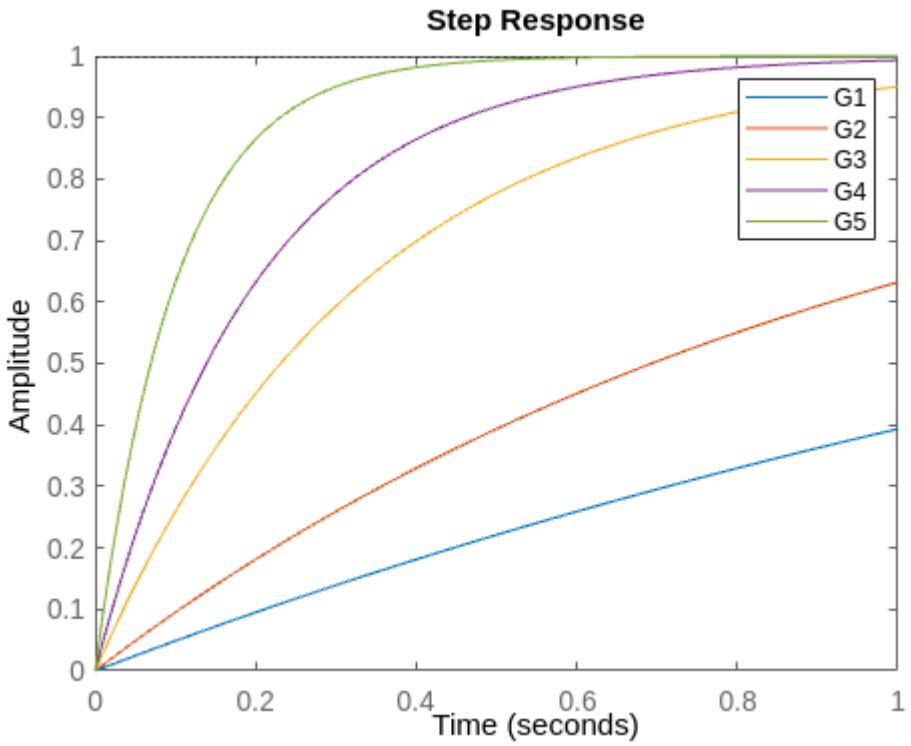
3. Realiza ahora una gráfica que incluya la respuesta al escalón unitario de todas ellas hasta un valor final de t = 10 s. Incluye la leyenda. Justifica el resultado obtenido.

```
figure; hold on
step(G1,G2,G3,G4,G5,10)
legend('G1','G2','G3','G4','G5'); hold off
```



4. Modifica la ganancia estática de cada una de las funciones anteriores para la respuesta al escalón de todas ellas tenga el mismo valor final igual a 1.

```
figure; hold on
step(G1*0.5,G2,G3*3,G4*5,G5*10,1)
legend('G1','G2','G3','G4','G5'); hold off
```




```
%ltiview(G) para ver los parámetros característicos del sistema
```

Ejercicio práctico 4. Respuesta temporal de sistemas de segundo orden

1. Construye cuatro funciones de transferencia de segundo orden de tal manera que todas tengan una K = 1, una frecuencia natural de 2 rad/s y una coeficiente de amortiguamiento igual a 0, 0.25, 1 y 5, respectivamente.

```
K=1; wn=2;
chi=[0 0.25 1 5];
G1 = tf(K*wn^2,[1 2*chi(1)*wn wn^2])
```

```
G1 =

     4
  -------
  s^2 + 4

Continuous-time transfer function.
```

```
G2 = tf(K*wn^2,[1 2*chi(2)*wn wn^2])
```

```
G2 =

       4
  -----------
  s^2 + s + 4

Continuous-time transfer function.
```

```
G3 = tf(K*wn^2,[1 2*chi(3)*wn wn^2])
```

```
G3 =

        4
  -------------
  s^2 + 4 s + 4

Continuous-time transfer function.
```

```
G4 = tf(K*wn^2,[1 2*chi(4)*wn wn^2])
```

```
G4 =

        4
  --------------
  s^2 + 20 s + 4

Continuous-time transfer function.
```

2. Extrae el valor numérico y realiza una única gráfica con la posición de los polos y ceros de las cuatro funciones de transferencia.

```
[polos1,ceros1] = pzmap(G1)
```

```
polos1 = 2×1 complex
   0.0000 + 2.0000i
   0.0000 - 2.0000i
ceros1 =

  0×1 empty double column vector
```

```
[polos2,ceros2] = pzmap(G2)
```

```
polos2 = 2×1 complex
  -0.5000 + 1.9365i
  -0.5000 - 1.9365i
ceros2 =

  0×1 empty double column vector
```

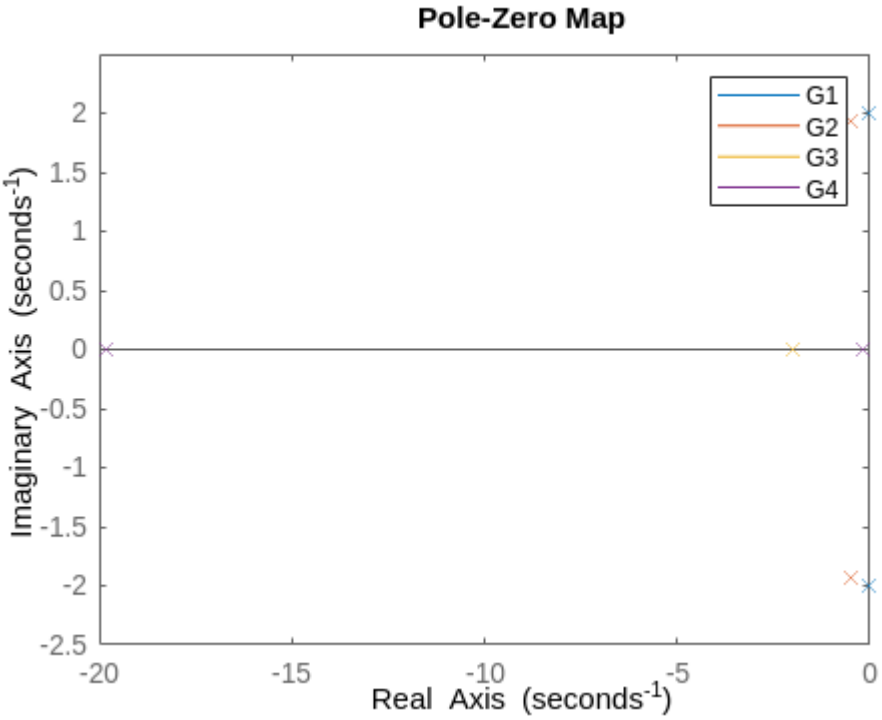
```
[polos3,ceros3] = pzmap(G3)
```

```
polos3 = 2×1
    -2
    -2
ceros3 =

  0×1 empty double column vector
```

```
[polos4,ceros4] = pzmap(G4)
```

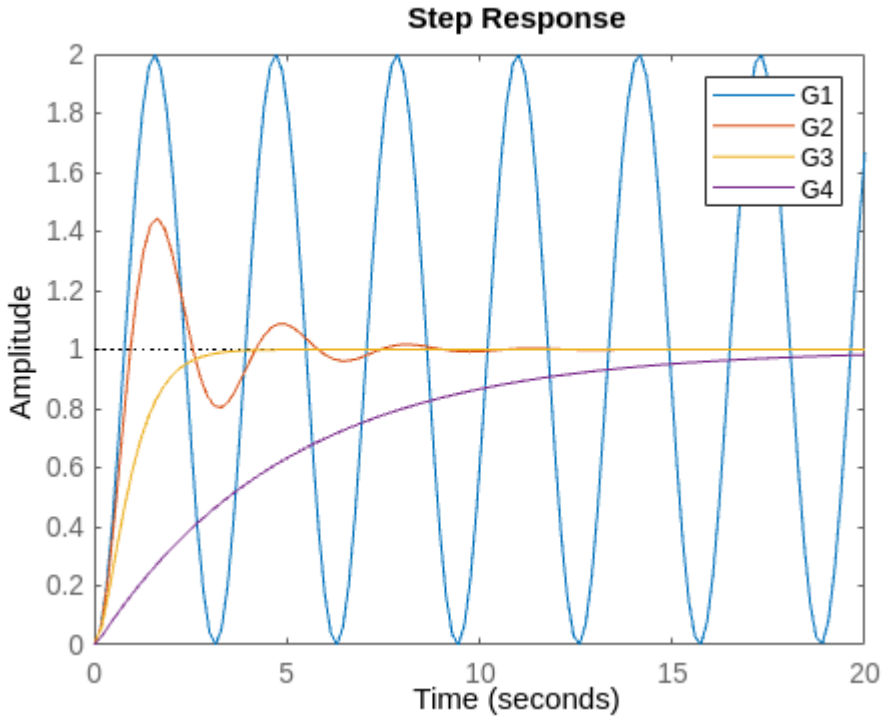
```
polos4 = 2×1
  -19.7980
   -0.2020
ceros4 =

  0×1 empty double column vector
```

```
pzmap(G1,G2,G3,G4)
legend('G1','G2','G3','G4');
```



3. Realiza una gráfica que incluya la respuesta al escalón unitario de todas ellas hasta un valor final que permita que todas las respuestas se vean con comodidad. Incluye la leyenda. Indica el tipo de cada una de las respuestas temporales (oscilatoria, subamortiguada, con amortiguamiento crítico o sobreamortiguada).

```
figure; hold on
step(G1,G2,G3,G4,20)
legend('G1','G2','G3','G4'); hold off
```



4. Calcula manualmente la posición de los polos para cada uno de los tipos de respuesta temporal al escalón, comprueba tus resultados y justifica las respuestas.

Nota: mediante el comando ord2() también se pueden definir funciones de transferencia de segundo orden. Consulta su sintaxis mediante doc ord2.

2.2 Extracción de parámetros característicos de la respuesta temporal

Acceder a los parámetros característicos:

respuesta_G = stepinfo(G)

Acceder a datos en un campo de array de estructura:

variable = structName.fieldName

Modificar ts y tr (ts = 0.05%, tr = 5-95%):

stepinfo(G, 'SettlingTimeThreshold', 0.005, 'RiseTimeThreshold', [0.05 0.95])

Ejercicio práctico 5. Extracción de los parámetros de la respuesta temporal

1. Extrae los valores numéricos del tiempo de subida y del tiempo de asentamiento para los sistemas de primer orden del ejercicio práctico 3, guardándolos en variables independientes. Comprueba dichos valores con los extraídos de la gráfica de la respuesta al escalón unitario.

```
G1 = zpk([],[-0.5],1);
A = stepinfo(G1);
tr1 = A.RiseTime
```

```
tr1 = 4.3940
```

```
ts1 = A.SettlingTime
```

```
ts1 = 7.8241
```

```
G2 = zpk([],[-1],1);
A = stepinfo(G1);
tr2 = A.RiseTime
```

```
tr2 = 4.3940
```

```
ts2 = A.SettlingTime
```

```
ts2 = 7.8241
```

```
G3 = zpk([],[-3],1);
A = stepinfo(G1);
tr3 = A.RiseTime
```

```
tr3 = 4.3940
```

```
ts3 = A.SettlingTime
```

```
ts3 = 7.8241
```

```
G4 = zpk([],[-5],1);
A = stepinfo(G1);
tr4 = A.RiseTime
```

```
tr4 = 4.3940
```

```
ts4 = A.SettlingTime
```

```
ts4 = 7.8241
```

```
G5 = zpk([],[-10],1);
A = stepinfo(G1);
tr5 = A.RiseTime
```

```
tr5 = 4.3940
```

```
ts5 = A.SettlingTime
```

```
ts5 = 7.8241
```

2. Repite el apartado anterior para las cuatro funciones de transferencia de sistemas de segundo orden especificadas en el ejercicio práctico 4.

```
k = 1; wn = 2;
chi = [0 0.25 1 5];
for t=1:4
    A = stepinfo(tf([k*wn^2],[1 2*chi(t)*wn wn^2]));
    ts(t) = A.SettlingTime;
    tr(t) = A.RiseTime;
end
ts
```

```
ts = 1×4
       NaN    7.0579    2.9170   19.4140
```

```
tr
```

```
tr = 1×4
       NaN    0.6343    1.6790   10.8747
```

3. Contrasta algunos tiempos de asentamiento con los valores teóricos esperados.

Ejercicio práctico 6. Diseño de sistemas de control

1. Utilizando comandos de control de flujo (for, while, if) realiza un script de MATLAB que calcule el valor que debe tener el polo de un sistema de primer orden para que su tiempo de subida (del 10 al 90%) sea igual a 1 ms.

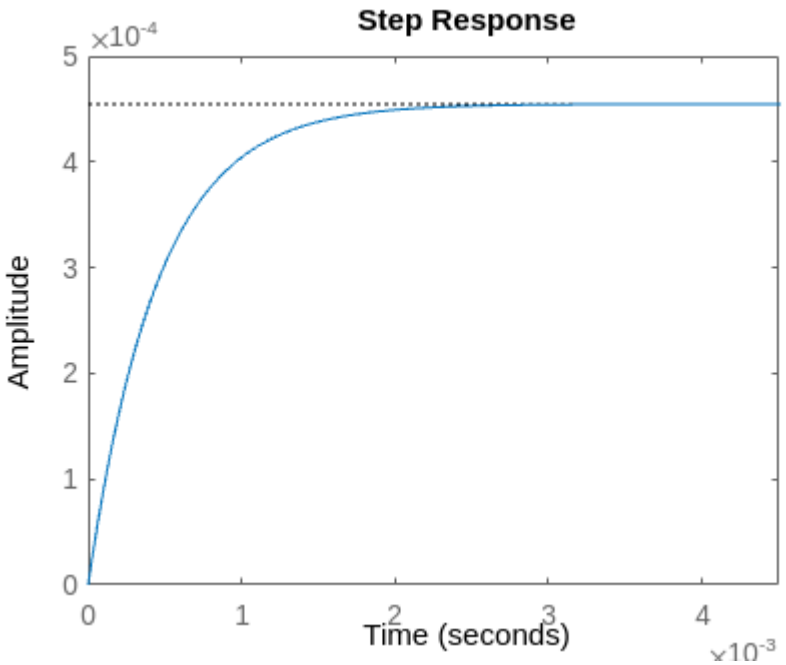
```
p=-1; tr=1; % Inicialización de variables
while tr > 0.001
    T = stepinfo(zpk([],[p],1));
    tr=T.RiseTime;
    p=p-1;
end
% Muestra de los resultados por pantalla
p
```

```
p = -2199
```

```
tr
```

```
tr = 9.9955e-04
```

```
step(zpk([],[p],1))
```



2. Repite el apartado anterior para extraer la frecuencia natural de un sistema de segundo orden con respuesta críticamente amortiguada que permite que el tiempo de asentamiento del sistema sea igual a 10 ms.

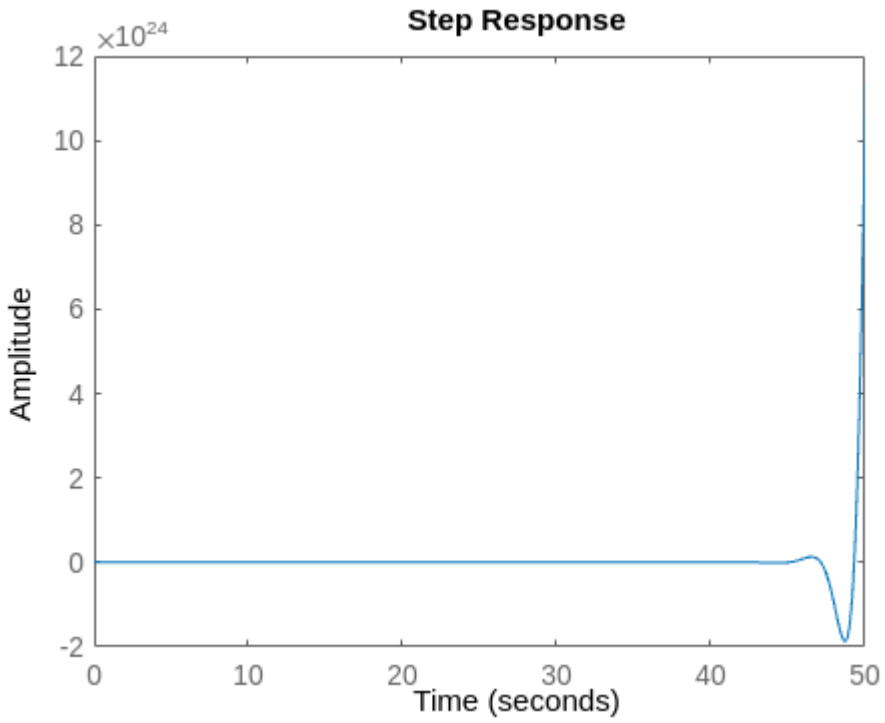
```
wn=1; ts=1;
while ts > 0.01
    [num,den] = ord2(wn,chi);
    T = stepinfo(tf(num,den));
    ts=T.SettlingTime;
    wn=wn+1;
end
wn
```

```
wn = 2
```

```
ts
```

```
ts = NaN
```

```
step(tf(num,den))
```



2.3 Respuesta a otro tipo de entradas (rampa unitaria, aceleración unitaria, etc ...)

Estudio de respuesta ante otras entradas:

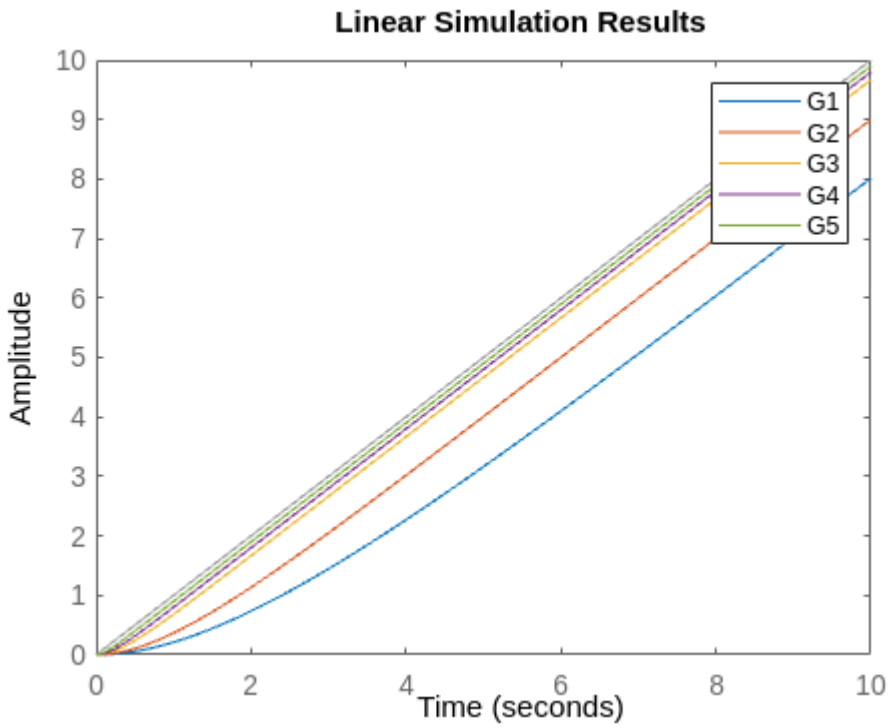
lsim(G,entrada,tiempo), donde:

tiempo = 0:dt:10 y entrada = tiempo .^2 (ejemplo).

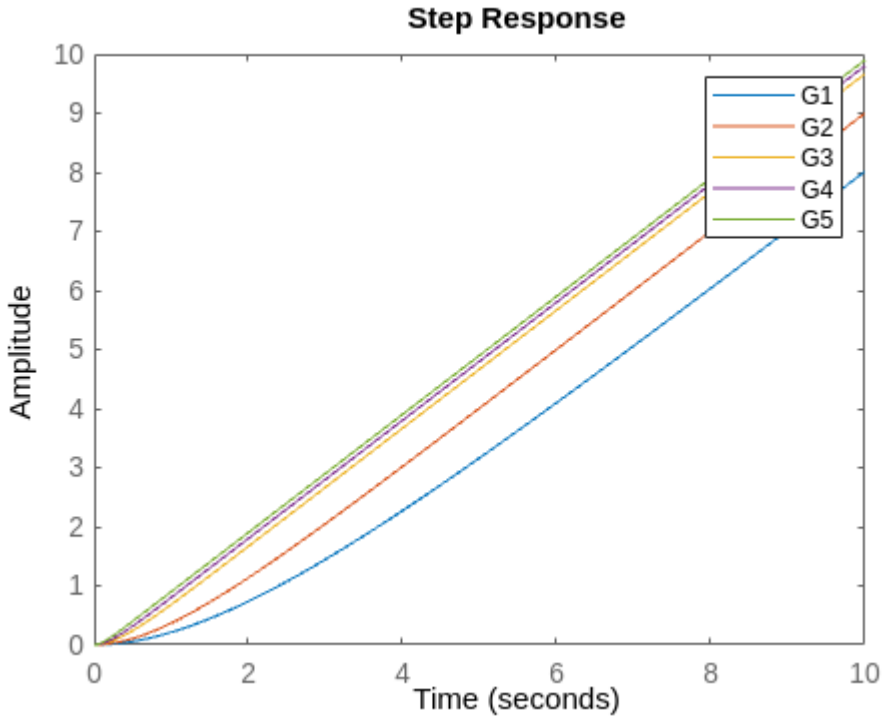
Ejercicio práctico 7. Respuesta de sistemas de 1º orden a entradas rampa y aceleración

1. Realiza una gráfica con la respuesta temporal de los sistemas de primer orden del ejercicio práctico 3 (una vez ajustadas las K) a una entrada rampa unitaria.

```
G1=zpk([],[-0.5],0.5);
G2=zpk([],[-1],1);
G3=zpk([],[-3],3);
G4=zpk([],[-5],5);
G5=zpk([],[-10],10);
% Resolución con lsim
t=linspace(0,10,100);
rampa=t;
figure; hold on
lsim(G1,rampa,t);
lsim(G2,rampa,t);
lsim(G3,rampa,t);
lsim(G4,rampa,t);
lsim(G5,rampa,t);
legend('G1','G2','G3','G4','G5'); hold off
```

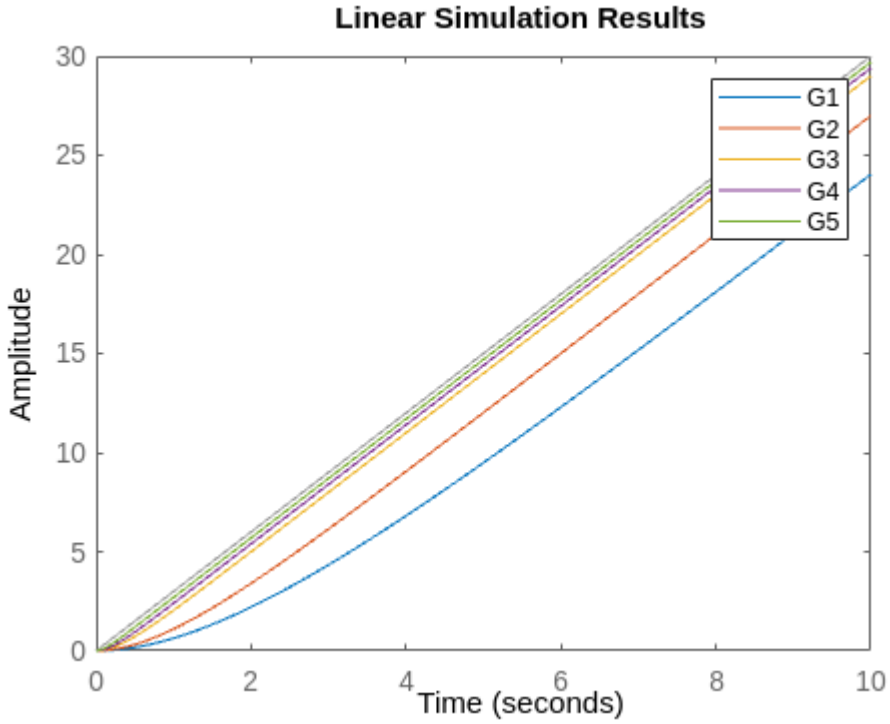



```
% Resolución con step
s = tf('s');
figure; hold on
step(G1/s,10);
step(G2/s,10);
step(G3/s,10);
step(G4/s,10);
step(G5/s,10);
legend('G1','G2','G3','G4','G5'); hold off
```



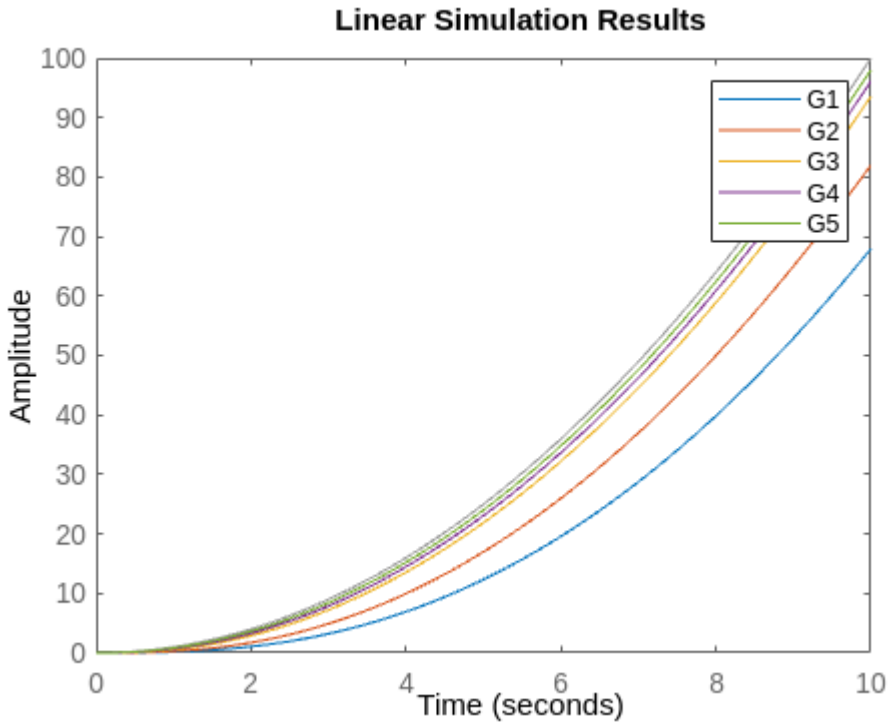
2. Repite el apartado anterior para una rampa de pendiente igual a 3.

```
rampa=3*t;
figure; hold on
lsim(G1,rampa,t);
lsim(G2,rampa,t);
lsim(G3,rampa,t);
lsim(G4,rampa,t);
lsim(G5,rampa,t);
legend('G1','G2','G3','G4','G5'); hold off
```



3. Realiza ahora una gráfica con la respuesta temporal de dichos sistemas a una entrada aceleración unitaria.

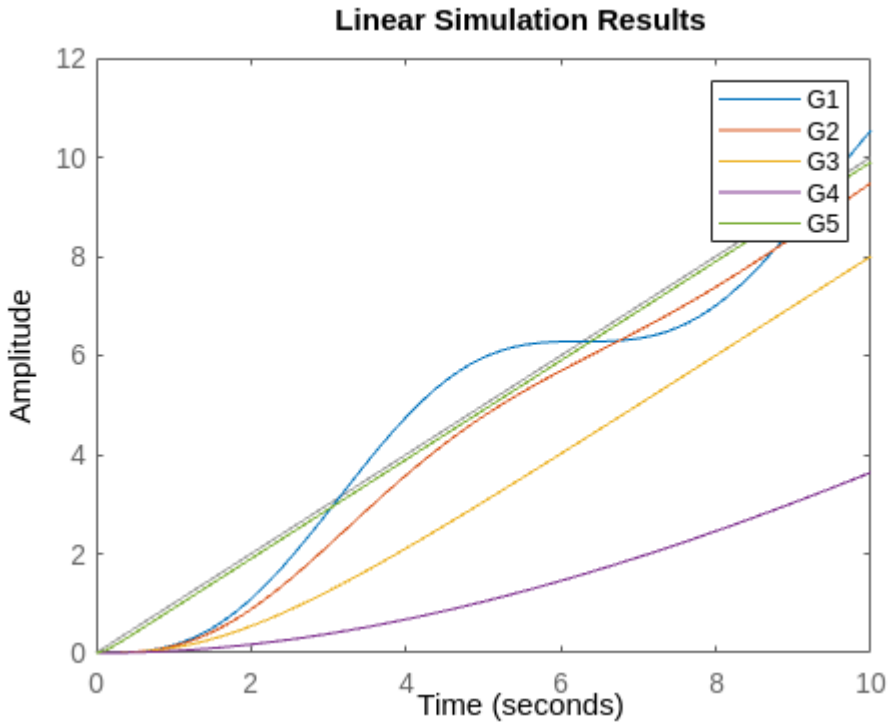
```
parabola=t.^2;
figure; hold on
lsim(G1,parabola,t);
lsim(G2,parabola,t);
lsim(G3,parabola,t);
lsim(G4,parabola,t);
lsim(G5,parabola,t);
legend('G1','G2','G3','G4','G5'); hold off
```



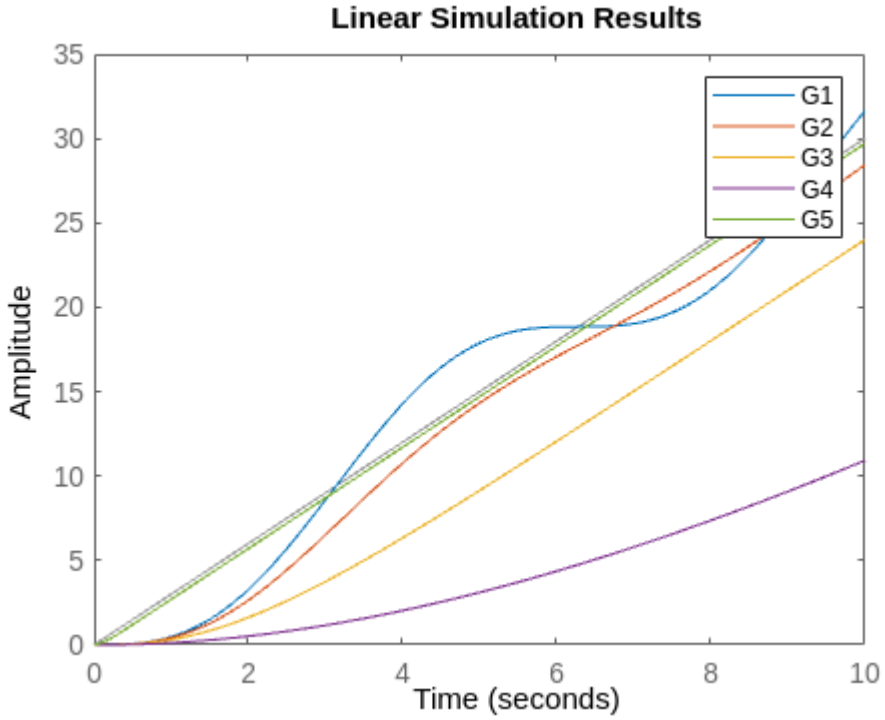
Ejercicio práctico 8. Respuesta de sistemas de 2º orden a entradas rampa y aceleración

1. Repite los tres puntos del apartado anterior para las cuatro funciones de transferencia de sistemas de segundo orden especificadas en el ejercicio práctico 4.

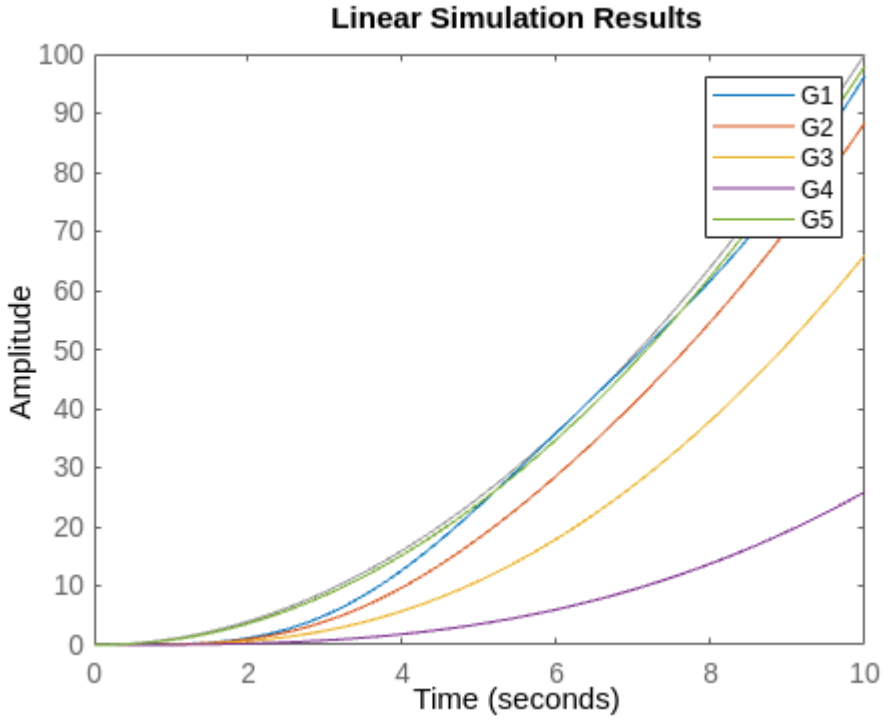
```
k=1; wn=1; chi=[0 0.25 1 5];
G1=tf([k*wn^2],[1 2*chi(1)*wn wn^2]);
G2=tf([k*wn^2],[1 2*chi(2)*wn wn^2]);
G3=tf([k*wn^2],[1 2*chi(3)*wn wn^2]);
G4=tf([k*wn^2],[1 2*chi(4)*wn wn^2]);
t=linspace(0,10,100);
% Apartado 1
rampa=t;
figure; hold on
lsim(G1,rampa,t);
lsim(G2,rampa,t);
lsim(G3,rampa,t);
lsim(G4,rampa,t);
lsim(G5,rampa,t);
legend('G1','G2','G3','G4','G5'); hold off
```




```
% Apartado 2
rampa=3*t;
figure; hold on
lsim(G1,rampa,t);
lsim(G2,rampa,t);
lsim(G3,rampa,t);
lsim(G4,rampa,t);
lsim(G5,rampa,t);
legend('G1','G2','G3','G4','G5'); hold off
```




```
% Apartado 3
parabola=t.^2;
figure; hold on
lsim(G1,parabola,t);
lsim(G2,parabola,t);
lsim(G3,parabola,t);
lsim(G4,parabola,t);
lsim(G5,parabola,t);
legend('G1','G2','G3','G4','G5'); hold off
```



3. Respuesta transitoria de sistemas de orden superior

3.1 Respuesta de un sistema de tercer orden (polos dominantes)

Ejercicio práctico 9. Respuesta de sistemas de 3º orden. Polos dominantes

1. Construye una función de transferencia de primer orden G1(s) donde la posición de su único polo sea configurable mediante la variable

$$G1(s) = 1/(1+s/p) = P/(s+P)$$

```
p=10;
G1=tf([p],[1 p])
```

```
G1 =

    10
  ------
  s + 10

Continuous-time transfer function.
```

2. Construye la siguiente función de transferencia de segundo orden G2(s). Extrae la posición (fija) de sus dos polos:

$$G2(s) = 10/s^2+2s+10$$

```
G2=tf([10],[1 2 10])
```

```
G2 =
```

```
       10
  --------------
  s^2 + 2 s + 10

Continuous-time transfer function.
```

```
[polos,ceros]=pzmap(G2)
```

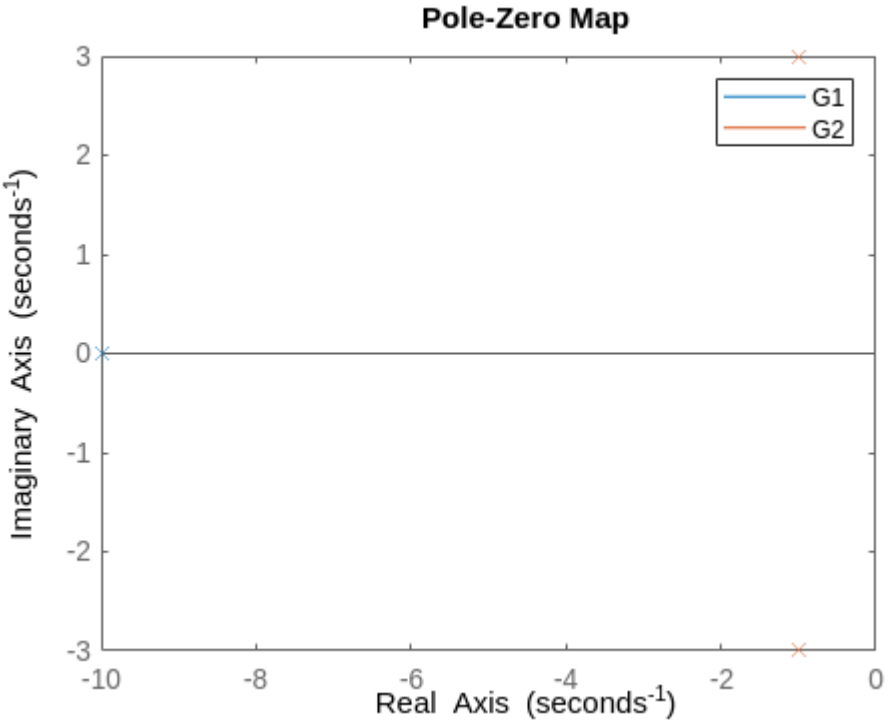
```
polos = 2×1 complex
  -1.0000 + 3.0000i
  -1.0000 - 3.0000i
ceros =

  0×1 empty double column vector
```

3. Define una nueva función de transferencia de tercer orden, G3(s), como el producto (cascada) de G1·G2. Realiza dos gráficas, por un lado una con posición de los polos de G1 y G2 (que serán los polos de G3) y por otro la respuesta al escalón de las tres funciones de transferencia, G1, G2 y G3. Activa la leyenda de la gráfica.

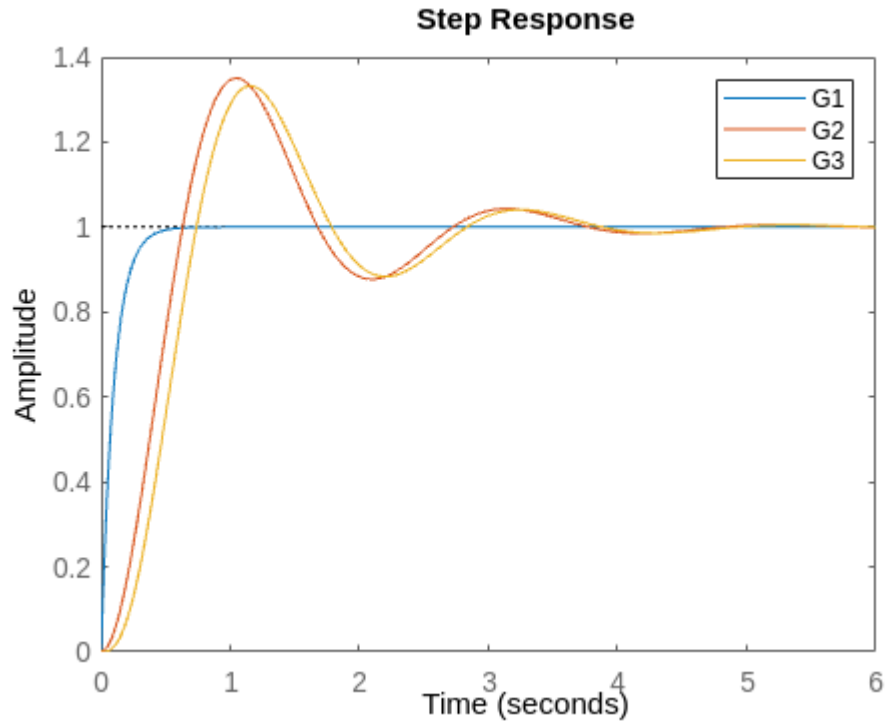
```
G3=series(G1,G2)
```

```
G3 =

             100
  ------------------------
  s^3 + 12 s^2 + 30 s + 100

Continuous-time transfer function.
```

```
% Mapa de polos y ceros
figure; hold on
pzmap(G1,G2);
legend('G1','G2'), hold off
```




```
% Respuesta temporal
figure; hold on
step(G1,G2,G3)
legend('G1','G2','G3'); hold off
```



4. Da valores a P desde 0,1 hasta 20 y relaciona la respuesta al escalón de G3 con las respuestas al escalón de G1 y G2, a través de la posición de sus polos. ¿Qué polos dominantes tiene G3 en cada caso? Razona tu respuesta.

5. ¿A partir de qué valor de P podrías decir que la respuesta de G3 es equivalente a la de G2, es decir, que el sistema de tercer orden se comporta como si tuviera sólo dos polos dominantes (los de G2)? Coteja tu respuesta en la bibliografía de la asignatura.

Ejercicio práctico 10. Respuesta de sistemas de 4º orden

1. En base al ejercicio anterior, añade a la función G3(s) del ejercicio anterior otro polo simple en posición s = -P2, construyendo una nueva función de 4º orden G4 = G1·G1b*G2 y comprueba si tus conclusiones del ejercicio anterior siguen siendo válidas para un sistema de 4º orden:

$$G1b(s) = 1/(1+s/P2) = P2/(s+P2)$$

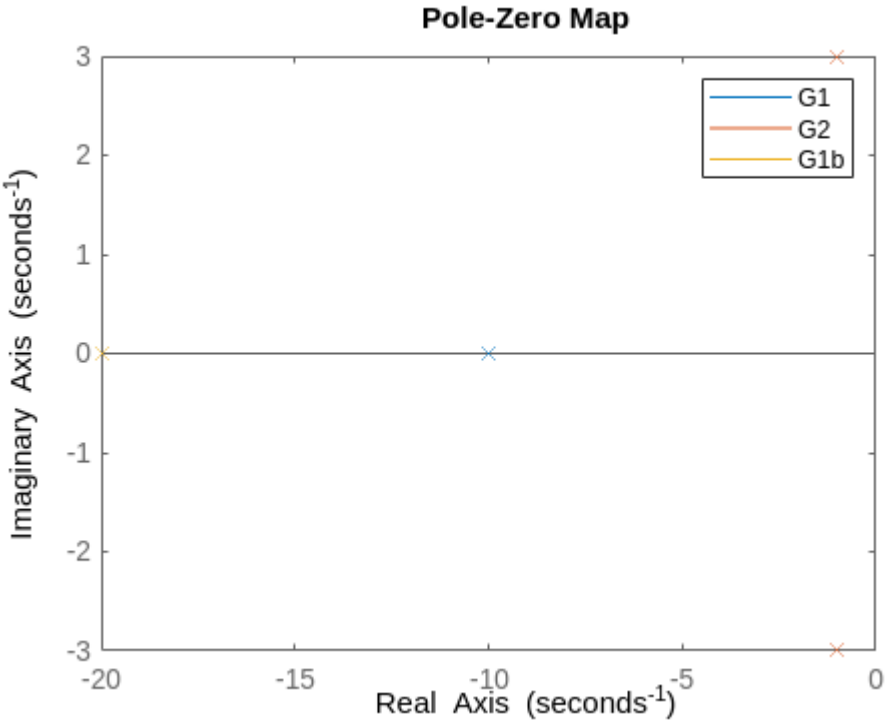
```
p2=20;
G1b=tf([p2],[1 p2])
```

```
G1b =

    20
  ------
  s + 20

Continuous-time transfer function.
```

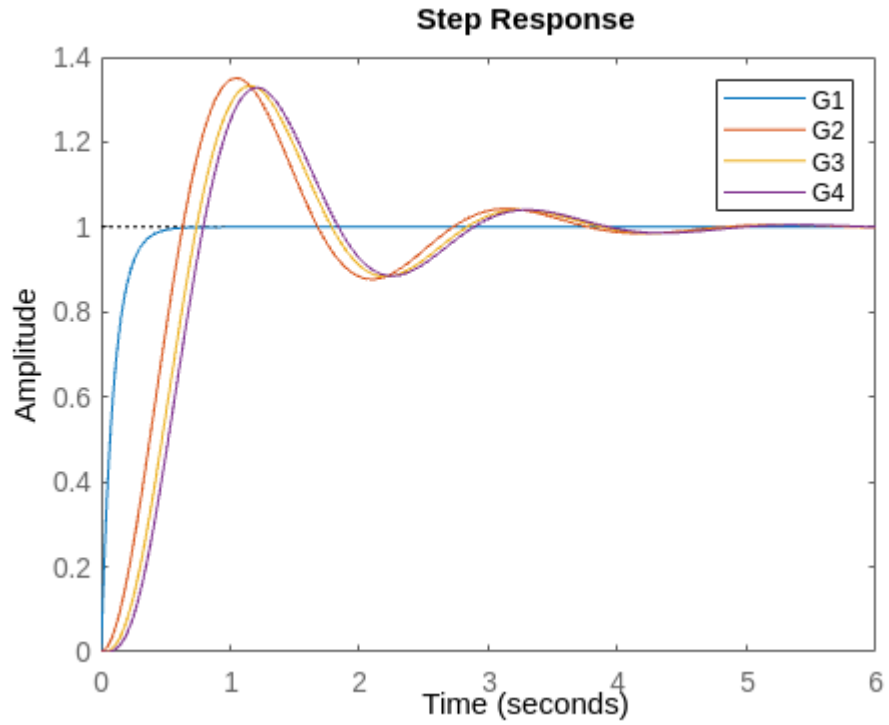
```
G4=series(G3,G1b)
```

```
G4 =

                 2000
  -------------------------------------
  s^4 + 32 s^3 + 270 s^2 + 700 s + 2000

Continuous-time transfer function.
```

```
% Mapa de polos y ceros
figure; hold on
pzmap(G1,G2,G1b);
legend('G1','G2','G1b'), hold off
```




```
% Respuesta temporal
figure; hold on
step(G1,G2,G3,G4)
legend('G1','G2','G3','G4'); hold off
```



4. Precisión y error en estado estacionario de un sistema de control

4.1 Error de posición en estado estacionario

Amplitud (configurable) del escalón:

Amplitud = 2

Almacenamos respuesta hasta t_final en el vector y:

[y,t] = step(Amplitud*GLC,t_final)

Error absoluto con su signo:

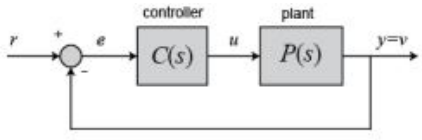
ess_pos_abs = Amplitud - y(end)

Error relativo en porcentaje con signo:

ess_pos_rel = 100*(Amplitud - y(end)) / Amplitud

Ejercicio práctico 11. Precisión y error de posición en estado estacionario

1. Construye el sistema en lazo cerrado de la figura, donde C(s) es igual a 1 y P(s) es un sistema sin ceros y con dos polos en s= -3 y s= -1.




```
C=1;
P=zpk([],[-3 -1],1);
```

2. ¿De qué TIPO es el sistema? Calcula su constante de error de posición y el error verdadero de posición en estado estacionario con las expresiones vistas en teoría.

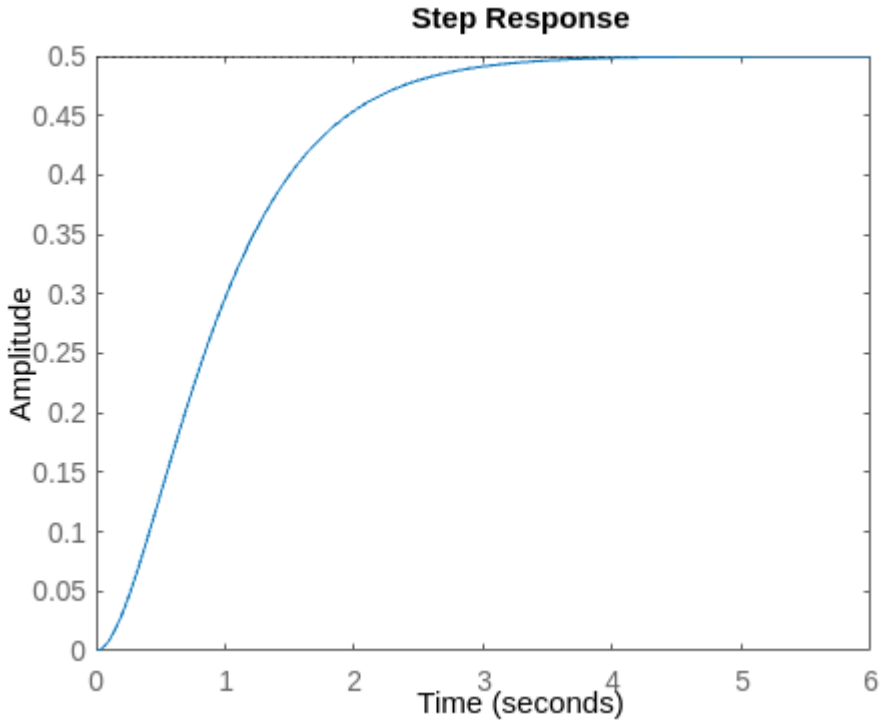
```
% Sistema tipo 0, ya que la función de transferencia en lazo abierto, no
% tiene polos en el origen
```

3. Comprueba dichos cálculos extrayendo los errores de posición absoluto y relativo utilizando MATLAB, mediante un escalón de amplitud igual a 2.

```
Glc=feedback(series(C,P),1)
```

```
Glc =

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

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

```
Amplitud=2;
figure; step(Amplitud*Glc)
```




```
[y,t]=step(Amplitud*Glc);
ess_pos_absoluto = Amplitud - y(end)
```

```
ess_pos_absoluto = 1.5002
```

```
ess_pos_relativo = (Amplitud - y(end))*100/Amplitud
```

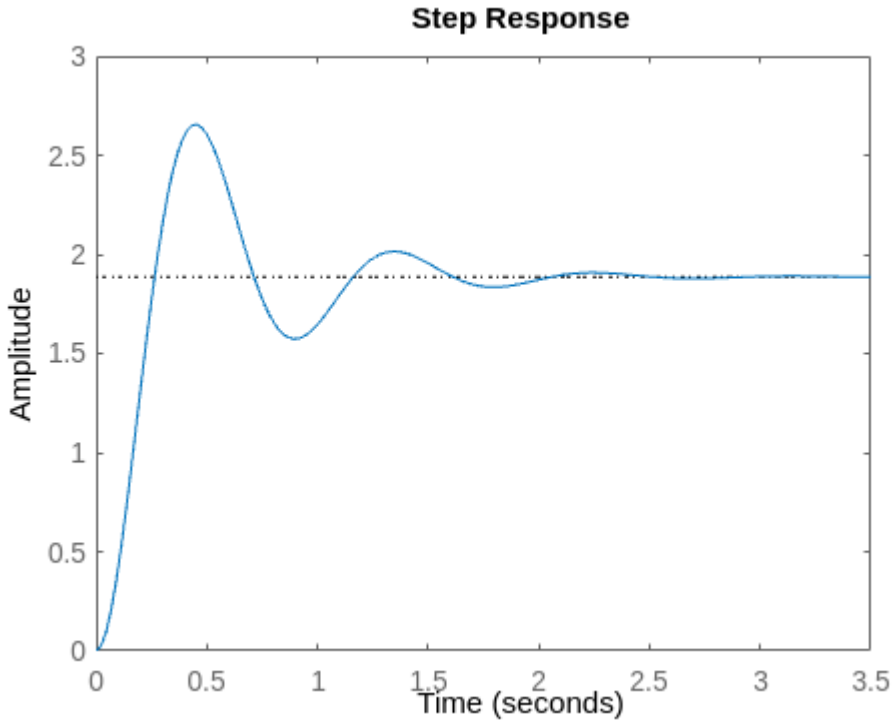
```
ess_pos_relativo = 75.0120
```

4. Repite el ejercicio asumiendo que ahora C(s) es igual a 50. ¿Ha cambiado el error de posición? ¿Se ha visto afectada la respuesta transitoria?

```
C=50;
Glc=feedback(series(C,P),1)
```

```
Glc =

        50
  ---------------
  (s^2 + 4s + 53)

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

```
figure; step(Amplitud*Glc)
```




```
[y,t]=step(Amplitud*Glc);
ess_pos_abs = Amplitud - y(end)
```

```
ess_pos_abs = 0.1098
```

```
ess_pos_rel = (Amplitud - y(end))*100/Amplitud
```

```
ess_pos_rel = 5.4899
```

4.2 Tipos de sistemas frente al error (error en estado estacionario de velocidad)

$$\lim_{s\to 0} G(s).H(s) = \lim_{s\to 0} \frac{K.\prod_{i=1}^{z}\left(\frac{s}{z_i}+1\right)}{s^n.\prod_{j=1}^{p}\left(\frac{s}{z_p}+1\right)} = \lim_{s\to 0}\frac{K}{s^n}$$

Valor de "n" = Tipo del sistema

El error verdadero del sistema depende de la entrada según su tipo, como se resume en la siguiente tabla:

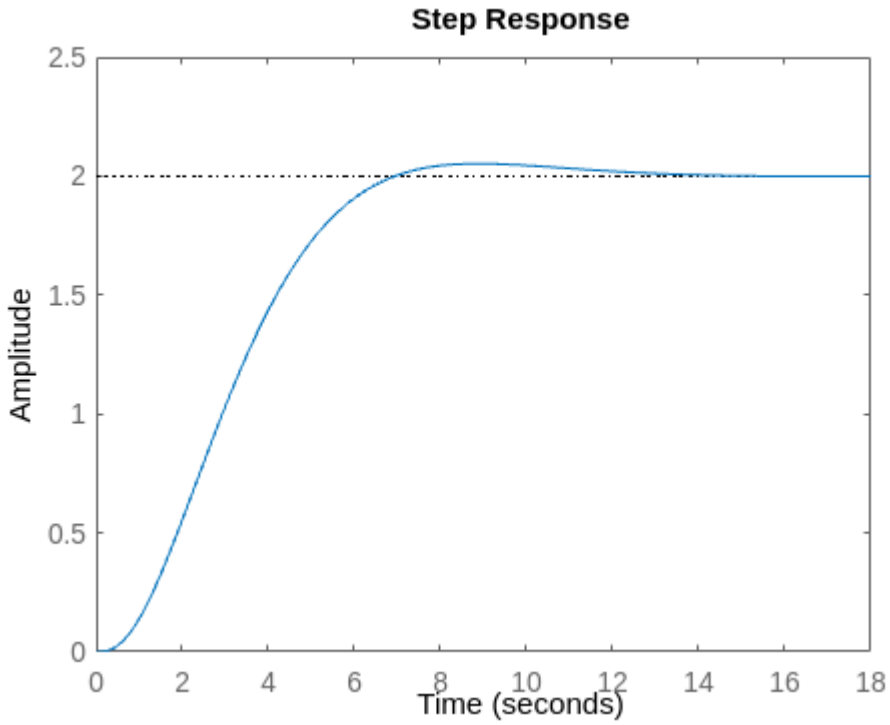
	Constantes			Error para diferentes entradas		
Tipo	k_p	k_v	k_a	Salto	Rampa	Parábola
0	K	0	0	M/(1+K)	∞	∞
1	∞	K	0	0	M/K	∞
2	∞	∞	K	0	0	M/K

Ejercicio práctico 12. Aumento del tipo de sistema frente al error.

1. Repite el ejercicio 11 introduciendo en C(s) las modificaciones necesarias que hagan aumentar en una unidad el TIPO del sistema frente al error.

```
C=tf([1],[1 0]);
```

```
Glc =

                 1
  ---------------------------------
  (s+3.148) (s^2 + 0.8521s + 0.3177)

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




```
ess_pos_abs = 0.0014
ess_pos_rel = 0.0693
```

2. Vuelve a calcular el error de posición con MATLAB y justifica el resultado obtenido.

```
Glc=feedback(series(C,P),1)
figure; step(Amplitud*Glc)
[y,t]=step(Amplitud*Glc);
ess_pos_abs = Amplitud - y(end)
ess_pos_rel = (Amplitud - y(end))*100/Amplitud
```

Ejercicio práctico 13. Precisión y error de velocidad en estado estacionario

1. Calcula ahora la constante de precisión de velocidad y el error verdadero en estado estacionario de velocidad del sistema que has definido ejercicio 12, mediante las expresiones teóricas.

2. Utilizando el comando “lsim” y la metodología de cálculo aprendida para los errores de posición, determina el error en estado estacionario de velocidad utilizando MATLAB.

3. Realiza la gráfica de la respuesta temporal del sistema ante una entrada de velocidad y explica los resultados obtenidos.

4. ¿En qué unidades crees que debe darse el error de velocidad absoluto?

```
C=tf(1,[1 0]);
P=zpk([],[-3 -1],1);
Glc=feedback(series(C,P),1)
t=linspace(0,10,100);
rampa=t;
figure; lsim(Glc,rampa,t)
[y,t]=lsim(Glc,rampa,t);
ess_vel_absoluto = rampa(end) - y(end)
ess_vel_relativo = (rampa(end) - y(end))*100/rampa(end)
kv = 1/ess_vel_absoluto;
```