UNIVERSIDAD DE VALPARAÍSO
Escuela de Ingeniería Civil Biomédica
CBM422 - Ingeniería de control automático

Tarea 1

Jorge Gonzalo Alejandro Alcaíno Brevis
Maximiliano Antonio Gaete Pizarro

1. Problema 1: Encuentre las transformadas inversas de Laplace, de las siguientes funciones

$$\begin{aligned} &\text{(a)} \quad F_1(s) = \frac{6s+3}{s^2} \\ &\text{(b)} \quad F_2(s) = \frac{5s+2}{(s+1)(s+2)^2} \end{aligned} \tag{0.1}$$

1.1. Parte (a):

$$F_1(s) = \frac{6s+3}{s^2}$$

Descomponiendo la función:

$$F_1(s) = 6 \cdot \frac{s}{s^2} + 3 \cdot \frac{1}{s^2}$$

La transformada inversa de Laplace de cada término es:

$$\mathcal{L}^{-1}\left\{\frac{s}{s^2}\right\} = 1, \quad \mathcal{L}^{-1}\left\{\frac{1}{s^2}\right\} = t$$

Por lo tanto, la transformada inversa de $F_1(s)$ es:

$$\mathcal{L}^{-1}\{F_1(s)\} = 6 + 3t$$

1.2. Parte (b):

$$F_2(s) = \frac{5s+2}{(s+1)(s+2)^2}$$

Descomponiendo en fracciones parciales:

$$F_2(s) = \frac{-3}{s+1} + \frac{3}{s+2} + \frac{8}{(s+2)^2}$$

Para esta fracción racional más compleja, podemos aplicar descomposición en fracciones parciales. La forma general sería:

$$\frac{5s+2}{(s+1)(s+2)^2} = \frac{A}{s+1} + \frac{B}{s+2} + \frac{C}{(s+2)^2}$$

Primero, multiplicamos ambos lados por $(s+1)(s+2)^2$ para eliminar los denominadores:

$$5s+2 = A(s+2)^2 + B(s+1)(s+2) + C(s+1)$$

Resolviendo en Python

```
from sympy import symbols, Eq, solve

# Definimos la variable s y los coeficientes A, B, C
s = symbols('s')
A, B, C = symbols('A␣B␣C')

# Expresion original
lhs = 5*s + 2

# Expansion en fracciones parciales
rhs = A*(s+2)**2 + B*(s+1)*(s+2) + C*(s+1)

# Expandimos el lado derecho
```

```
expanded_rhs = rhs.expand()

# Igualamos ambos lados para resolver el sistema de ecuaciones
equations = Eq(lhs, expanded_rhs)

# Resolvemos para A, B y C
coefficients = solve(equations, [A, B, C])
coefficients
```

Expandiendo ambos lados y resolviendo para A, B, y C, encontramos los coeficientes que necesitamos. Esto nos da la forma correcta para aplicar la transformada inversa de Laplace a cada término:

$$\frac{5s+2}{(s+1)(s+2)^2} = \frac{-3}{s+1} + \frac{3}{s+2} + \frac{8}{(s+2)^2}$$

Aplicando la transformada inversa de Laplace a cada término:

$$\mathcal{L}^{-1}\left\{\frac{1}{s+1}\right\} = e^{-t}, \quad \mathcal{L}^{-1}\left\{\frac{1}{s+2}\right\} = e^{-2t}, \quad \mathcal{L}^{-1}\left\{\frac{1}{(s+2)^2}\right\} = te^{-2t}$$

Por lo tanto, la transformada inversa de $F_2(s)$ es:

$$\mathcal{L}^{-1}\{F_2(s)\} = -3e^{-t} + 3e^{-2t} + 8te^{-2t}$$

2. Problema 2: Obtenga la Función de Transferencia del sistema definido por las ecuaciones

$$\begin{aligned}\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix} &= \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -2 & -4 & -6\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}\\ \begin{bmatrix}y_1\\ y_2\end{bmatrix} &= \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}\end{aligned} \tag{0.2}$$

2.1. Obtención de la función de transferencia del sistema:

Dado el sistema de ecuaciones en el espacio de estados:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

donde las matrices A, B, C y D están definidas como:

$$A = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -2 & -4 & -6\end{bmatrix}, \quad B = \begin{bmatrix}0 & 0\\ 0 & 1\\ 1 & 0\end{bmatrix}, \quad C = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}, \quad D = \begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

La función de transferencia $\mathbf{G}(s)$ en el dominio de Laplace se calcula utilizando la fórmula:

$$\mathbf{G}(s) = C(sI - A)^{-1}B + D$$

Paso 1: Calcular $(sI - A)$
La matriz identidad I es:

$$I = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

Por lo tanto, $(sI - A)$ es:

$$sI - A = \begin{bmatrix}s & -1 & 0\\ 0 & s & -1\\ 2 & 4 & s+6\end{bmatrix}$$

Paso 2: Inversa de $(sI - A)$
Ahora, calculamos la inversa de $(sI - A)$:
Resolviendo en python

```
from sympy import Matrix, eye, symbols, simplify

# Definir las matrices y la variable s
s = symbols('s')
A = Matrix([[0, 1, 0], [0, 0, 1], [-2, -4, -6]])
B = Matrix([[0, 0], [0, 1], [1, 0]])
C = Matrix([[1, 0, 0], [0, 1, 0]])
D = Matrix([[0, 0], [0, 0]])
I = eye(3)  # Matriz identidad de 3x3

# Calcular (sI - A) y su inversa
sI_A_inv = (s * I - A).inv()
```

$$(sI - A)^{-1} = \frac{1}{s^3 + 6s^2 + 4s + 2} \begin{bmatrix} s^2 + 6s + 4 & s + 6 & 1 \\ 2s + 4 & s^2 + 6s + 2 & s + 6 \\ 2 & 2s + 4 & s^2 + 6s \end{bmatrix}$$

Paso 3: Multiplicación con B
Multiplicamos $(sI - A)^{-1}$ por B:

$$(sI - A)^{-1}B = \frac{1}{s^3 + 6s^2 + 4s + 2} \begin{bmatrix} 1 & s + 6 \\ s & s(s + 6) \end{bmatrix}$$

Paso 4: Multiplicación con C y adición de D
Finalmente, multiplicamos por C y sumamos D, obteniendo la función de transferencia:
Resolviendo en Python

```
# Calcular la funcion de transferencia G(s) = C(sI - A)^(-1)B + D
G_s = simplify(C * sI_A_inv * B + D)
G_s
```

$$\mathbf{G}(s) = \begin{bmatrix} \frac{1}{s^3+6s^2+4s+2} & \frac{s+6}{s^3+6s^2+4s+2} \\ \frac{s}{s^3+6s^2+4s+2} & \frac{s(s+6)}{s^3+6s^2+4s+2} \end{bmatrix}$$

Esta es la función de transferencia del sistema, que describe la relación entre las entradas u_1 y u_2, y las salidas y_1 y y_2.

3. Problema 3: Mediante la simplificación del diagrama de bloques de la Figura 1, obtenga las siguientes funciones de transferencia

$$\left.\frac{Y(s)}{R(s)}\right|_{N=0} \qquad \left.\frac{Y(s)}{N(s)}\right|_{R=0}$$

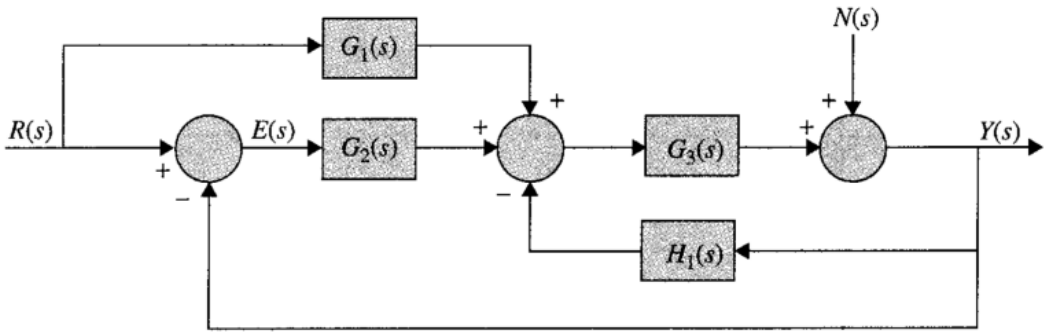


Figura 1: Diagrama de bloques del sistema

4. Problema 4: Obtenga la Función de Transferencia Eo(s)/Ei(s) del circuito eléctrico RLC que se muestra en la Figura 2

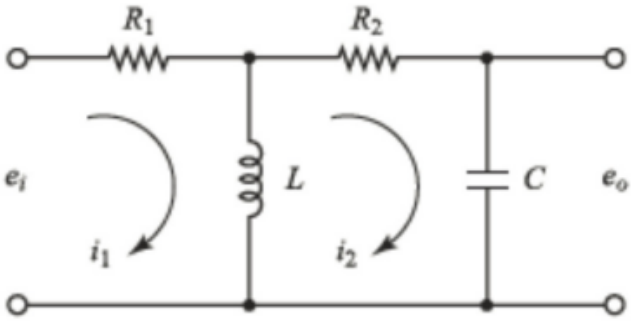


Figura 2: Circuito eléctrico RLC

5. Problema 5: Considere el sistema de la Figura 3(a). El factor de amortiguamiento relativo ζ del sistema es igual a 0,158, y la frecuencia natural no amortiguada ω_n es de 3,16 rad/seg. Para mejorar la estabilidad se emplea una segunda retroalimentación, como se muestra en la Figura 3(b). Determine el valor de K_h, para que el factor de amortiguamiento relativo del sistema sea ahora 0,5. Obtenga la curva de respuesta al escalón unitario (analítica y computacionalmente), tanto del sistema original como del sistema con la nueva retroalimentación. Y analice.

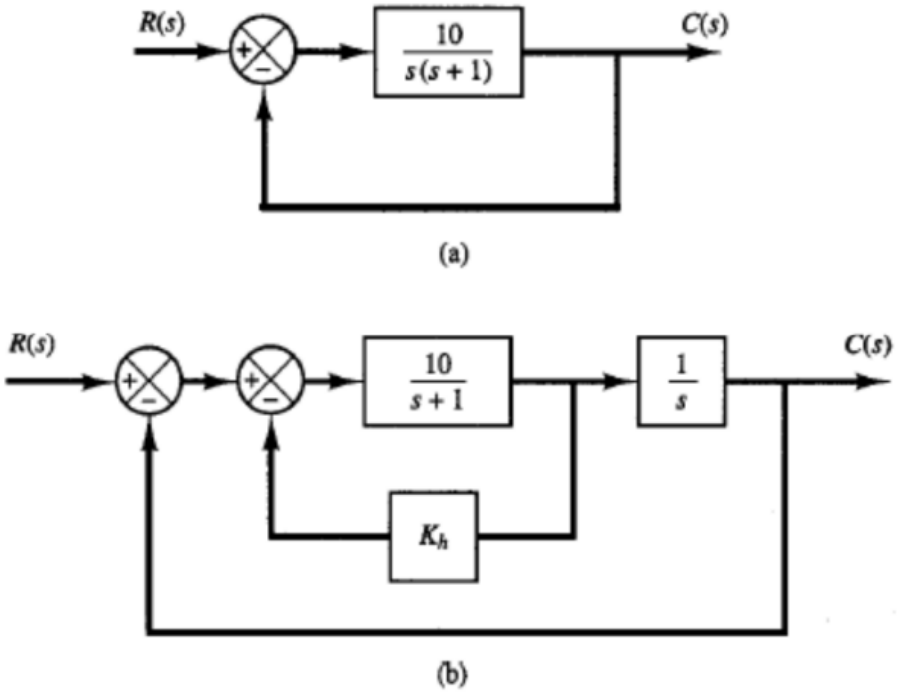


Figura 3: Sistema de la Figura 3(a) y 3(b)

5.1. El sistema dado en la Figura 3(a) tiene la siguiente función de transferencia:

$$G(s) = \frac{10}{s(s+1)}$$

Con una retroalimentación unitaria, la función de transferencia de lazo cerrado es:

$$T(s) = \frac{G(s)}{1+G(s)} = \frac{\frac{10}{s(s+1)}}{1+\frac{10}{s(s+1)}} = \frac{10}{s^2+s+10}$$

De acuerdo a los datos proporcionados:

- El factor de amortiguamiento relativo (ζ) es 0,158.

- La frecuencia natural no amortiguada (ω_n) es 3,16 rad/seg.

Estos parámetros corresponden a una ecuación característica de segundo orden de la forma:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

Sustituyendo los valores:

$$s^2 + 2(0{,}158)(3{,}16)s + (3{,}16)^2 = s^2 + 0{,}998s + 9{,}9856$$

Lo cual se aproxima bastante a la ecuación característica del sistema de lazo cerrado original: $s^2 + s + 10$.
Implementación computacional en Python

```
import control as ctrl
import matplotlib.pyplot as plt

# Sistema original
num_orig = [10]
den_orig = [1, 1, 10]
sys_orig = ctrl.TransferFunction(num_orig, den_orig)

# Simulacion de la respuesta al escalon
t, y_orig = ctrl.step_response(sys_orig)

# Graficar resultados
plt.plot(t, y_orig, color='b', label='Sistema␣Original')
plt.title('Respuesta␣al␣escalon␣unitario')
plt.xlabel('Tiempo␣(s)')
plt.ylabel('Respuesta')
plt.legend()
plt.grid()
plt.show()
```

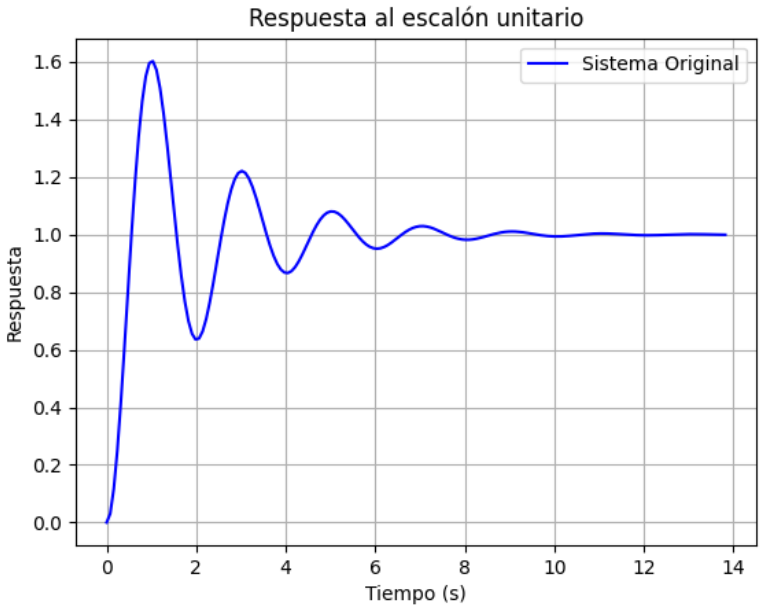


Figura 4: Respuesta al escalón unitario del sistema original

5.2. Sistema con nueva retroalimentación (Figura 3b)

En la Figura 3(b), se añade una segunda retroalimentación con ganancia K_h. Esto modifica la ecuación característica del sistema.

La nueva función de transferencia con K_h será:

$$T_{\text{nuevo}}(s) = \frac{10}{s(s+1) + K_h}$$

El nuevo polinomio característico del sistema en lazo cerrado es:

$$s^2 + s + 10K_h = 0$$

Comparando el polinomio característico del sistema con la forma estándar:

$$s^2 + s + 10K_h = s^2 + 2\zeta\omega_n s + \omega_n^2$$

De esta comparación, obtenemos dos ecuaciones:

- Para el término de s: $2\zeta\omega_n = 1$
- Para el término constante: $\omega_n^2 = 10K_h$

Cálculo de ω_n

Usamos la primera ecuación $2\zeta\omega_n = 1$ para despejar ω_n:

$$\omega_n = \frac{1}{2\zeta} = \frac{1}{2(0{,}5)} = 1$$

Cálculo de K_h

Sustituimos el valor de $\omega_n^2 = 1^2 = 1$ en la segunda ecuación $\omega_n^2 = 10K_h$:

$$1 = 10K_h$$

$$K_h = \frac{1}{10} = 0{,}1$$

Implementación computacional en Python

```
import control as ctrl
import matplotlib.pyplot as plt

# Sistema con nueva retroalimentacion
K_h = 0.1
num_new = [10]
den_new = [1, 1, 10*K_h]
sys_new = ctrl.TransferFunction(num_new, den_new)

t2, y_new = ctrl.step_response(sys_new)

# Graficar resultados
plt.plot(t2, y_new, color='r', label='Sistema con nueva retroalimentacion')
plt.title('Respuesta al escalon unitario')
plt.xlabel('Tiempo (s)')
plt.ylabel('Respuesta')
plt.legend()
plt.grid()
plt.show()
```

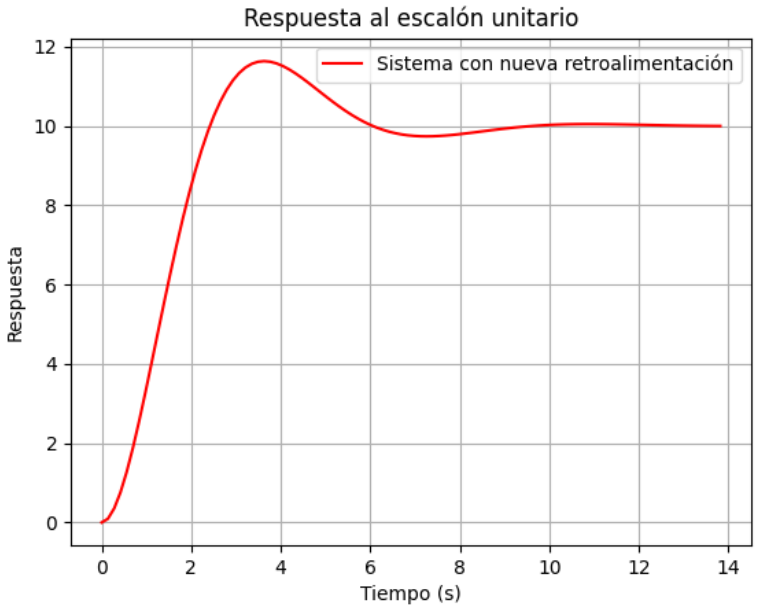


Figura 5: Respuesta al escalón unitario del sistema con nueva retroalimentación

5.3. Respuesta al escalón unitario

Sistema original

La función de transferencia del sistema original es:

$$T(s) = \frac{10}{s^2 + s + 10}$$

Usando la transformada inversa de Laplace para la respuesta al escalón unitario, obtenemos una respuesta típica de un sistema de segundo orden subamortiguado, con un factor de amortiguamiento $\zeta = 0{,}158$.

Sistema con retroalimentación

Para el sistema con la segunda retroalimentación, la función de transferencia se convierte en:

$$T_{\text{nuevo}}(s) = \frac{10}{s^2 + s + 1}$$

Este sistema tendrá una mejor estabilidad debido a su mayor factor de amortiguamiento ($\zeta = 0{,}5$), lo que reducirá las oscilaciones en la respuesta al escalón.

Análisis

El nuevo sistema con $K_h = 0{,}1$ tendrá una respuesta al escalón más rápida y con menos sobreoscilación en comparación con el sistema original, que tiene un amortiguamiento mucho menor. La mejora en la estabilidad es evidente al aumentar el factor de amortiguamiento de 0,158 a 0,5.

Implementación computacional en Python

```
import control as ctrl
import matplotlib.pyplot as plt

# Sistema original
num_orig = [10]
den_orig = [1, 1, 10]
```

```
sys_orig = ctrl.TransferFunction(num_orig, den_orig)

# Sistema con nueva retroalimentacion
K_h = 0.1
num_new = [10]
den_new = [1, 1, 10*K_h]
sys_new = ctrl.TransferFunction(num_new, den_new)

# Simulacion de la respuesta al escalon
t, y_orig = ctrl.step_response(sys_orig)
t2, y_new = ctrl.step_response(sys_new)

# Graficar resultados
plt.plot(t, y_orig, color='b', label='Sistema Original')
plt.plot(t2, y_new, color='r', label='Sistema con nueva retroalimentacion')
plt.title('Respuesta al escalon unitario')
plt.xlabel('Tiempo (s)')
plt.ylabel('Respuesta')
plt.legend()
plt.grid()
plt.show()
```

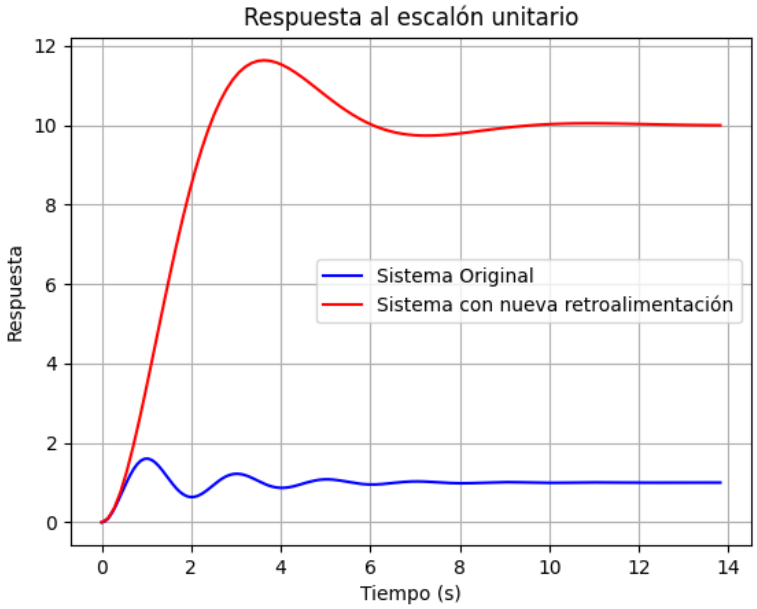


Figura 6: Respuesta al escalón unitario de ambos sistemas

6. Problema 6: Considerando el sistema de la Figura 7, determine el valor de k , de modo que el factor de amortiguamiento sea 0,5. Basado en esto, obtenga el tiempo de levantamiento tr , el tiempo peak tp , el sobrepaso máximo Mp , y el tiempo de asentamiento ts , en la respuesta al escalón unitario. Obtenga analíticamente y muestre la respuesta en el tiempo (computacionalmente).

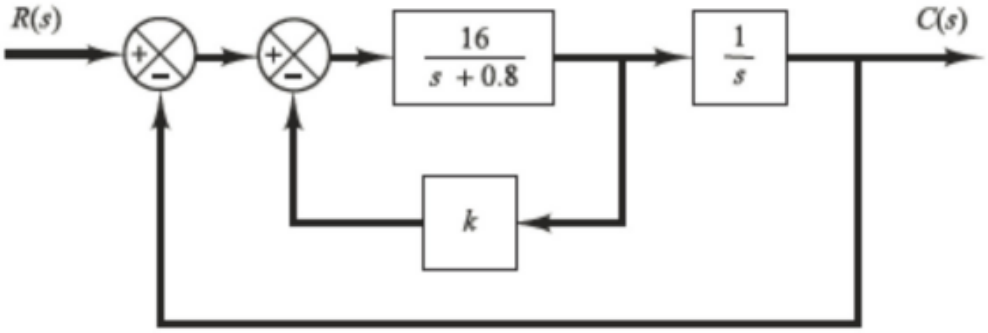


Figura 7: Sistema de la Figura 7

La función de transferencia del sistema en la Figura 7 es:

$$G(s) = \frac{16}{s + 0{,}8}$$

La segunda retroalimentación proporcional tiene una ganancia k. Para encontrar la función de transferencia total en lazo cerrado, debemos considerar ambas retroalimentaciones. La ecuación característica del sistema en lazo cerrado es:

$$T(s) = \frac{G(s)}{1 + G(s) \cdot k \cdot \frac{1}{s}} = \frac{\frac{16}{s+0{,}8}}{1 + \frac{16k}{s(s+0{,}8)}} = \frac{16}{s(s + 0{,}8) + 16k}$$

6.1. Determinación del valor de k

Para un sistema de segundo orden, la ecuación característica estándar es:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

Donde:

- $\zeta = 0{,}5$ (factor de amortiguamiento relativo),
- ω_n es la frecuencia natural no amortiguada.

La ecuación característica del sistema será:

$$s^2 + 0{,}8s + 16k = 0$$

El factor de amortiguamiento está relacionado con la frecuencia natural y la constante del término lineal en s por la fórmula:

$$2\zeta\omega_n = 0{,}8$$

Sustituyendo $\zeta = 0{,}5$:

$$2(0{,}5)\omega_n = 0{,}8 \quad \Rightarrow \quad \omega_n = \frac{0{,}8}{1} = 0{,}8$$

La frecuencia natural ω_n también está relacionada con el término constante en la ecuación característica:

$$\omega_n^2 = 16k$$

Sustituyendo $\omega_n = 0{,}8$:

$$0{,}8^2 = 16k \quad \Rightarrow \quad k = \frac{0{,}64}{16} = 0{,}04$$

6.2. Cálculo de parámetros de la respuesta al escalón unitario

6.2.1. Tiempo de levantamiento t_r

El tiempo de levantamiento es el tiempo que tarda la salida en pasar del 10 % al 90 % del valor final. Para un sistema subamortiguado con $\zeta = 0{,}5$, una aproximación es:

$$t_r \approx \frac{1{,}8}{\omega_n} = \frac{1{,}8}{0{,}8} = 2{,}25\,\text{segundos}$$

6.2.2. Tiempo peak t_p

El tiempo peak es el tiempo que tarda la respuesta en alcanzar su primer máximo. Para un sistema de segundo orden:

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} = \frac{\pi}{0{,}8\sqrt{1-0{,}5^2}} = \frac{\pi}{0{,}8 \times 0{,}866} \approx 4{,}55\,\text{segundos}$$

6.2.3. Sobrepaso máximo M_p

El sobrepaso máximo se puede calcular con la siguiente fórmula para un sistema de segundo orden:

$$M_p = e^{\left(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}\right)} = e^{\left(\frac{-\pi(0{,}5)}{\sqrt{1-(0{,}5)^2}}\right)} = e^{\left(\frac{-1{,}57}{0{,}866}\right)} \approx 16{,}3\,\%$$

6.2.4. Tiempo de asentamiento t_s

El tiempo de asentamiento es el tiempo que tarda la respuesta en quedarse dentro de un 2 % del valor final. Para un sistema de segundo orden, el tiempo de asentamiento se aproxima como:

$$t_s \approx \frac{4}{\zeta\omega_n} = \frac{4}{0{,}5 \times 0{,}8} = 10\,\text{segundos}$$

6.3. Simulación computacional

Implementación computacional en Python

```
import control as ctrl
import matplotlib.pyplot as plt

# Parametros del sistema
k = 0.04  # Valor de k determinado
num = [16]
den = [1, 0.8, 16*k]

# Crear el sistema en lazo cerrado
```

```
sys = ctrl.TransferFunction(num, den)

# Simulacion de la respuesta al escalon
t, y = ctrl.step_response(sys)

# Graficar la respuesta

plt.plot(t, y, label='Respuesta␣al␣escalon␣unitario')
plt.title('Respuesta␣al␣escalon␣unitario␣(k␣=␣0.04)')
plt.xlabel('Tiempo␣(s)')
plt.ylabel('Respuesta')
plt.axvline(x=2.25, color='red', linestyle='--', label='Tiempo␣de␣levantamiento␣(Tr)
    ␣2.25␣[s]')
plt.axvline(x=4.55, color='green', linestyle='--', label='Tiempo␣peak␣(Tp)␣4.55␣[s]'
    )
plt.axvline(x=10, color='purple', linestyle='--', label='Tiempo␣de␣asentamiento␣(Ts)
    ␣10␣[s]')
plt.plot([], [], '␣', label='Sobrepaso␣16.3%')
plt.grid()
plt.legend()
plt.show()
```

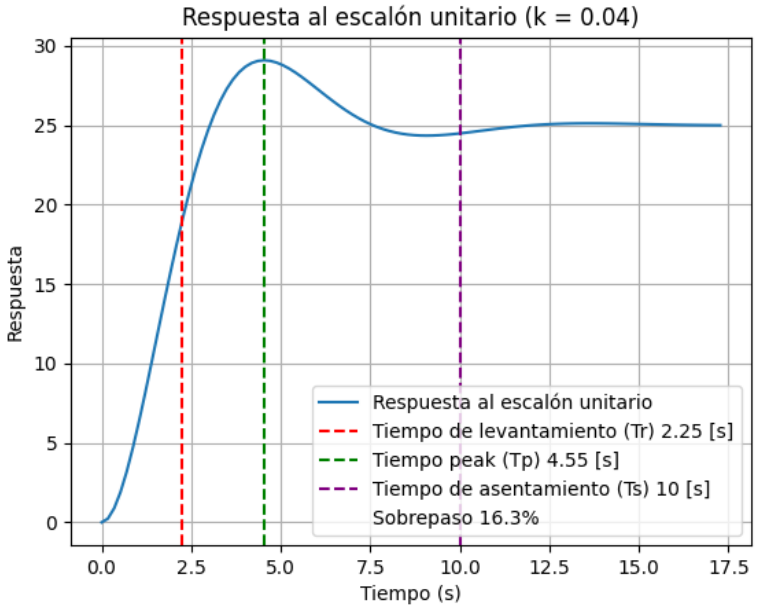
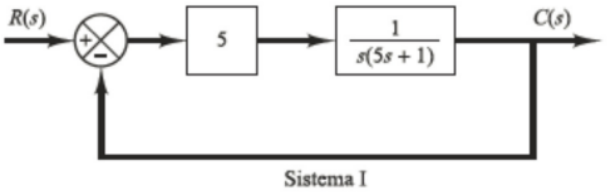


Figura 8: Respuesta al escalón unitario del sistema con $k = 0{,}04$

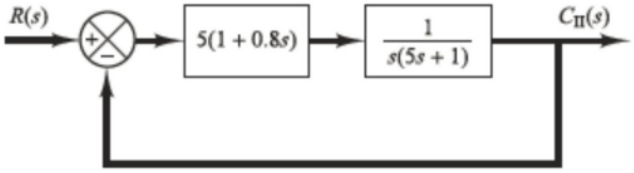
Este código genera la respuesta al escalón unitario y te permite analizar visualmente cómo se comporta el sistema con el valor $k = 0{,}04$.

7. Problema 7: Las Figuras 9 muestran tres sistemas. El sistema I es un sistema de control de posición; el sistema II es un sistema de control de posición con un control PD; y el sistema III es un sistema de control de posición con retroalimentación de velocidad. Compare las respuestas a un escalón unitario e impulso unitario de los tres sistemas (computacionalmente, y ojalá en una misma gráfica). ¿Qué sistema es mejor con respecto a la velocidad de respuesta, y sobre el sobrepaso (elongación) máxima, en la respuesta al escalón?



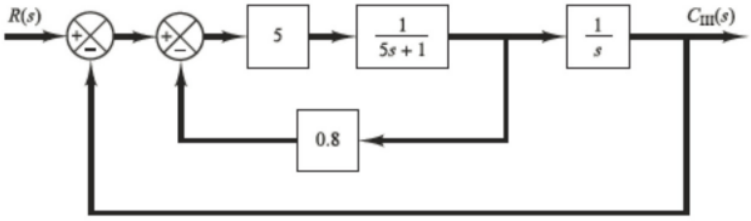
Sistema I

Figura (a)



Sistema II

Figura (b)



Sistema III

Figura (c)

Figura 9: Sistemas I, II y III