UNIVERSIDAD
DE LA FRONTERA

Universidad de La Frontera
Departamento de Ingeniería Eléctrica

IIE339 - Herramientas de Análisis de Señales

Tarea N º 2

Nombre: Maximiliano Ortúzar
Matricula: 21631361523
Carrera: Ingeniería Civil Electrónica
Profesor: Dr. Fernando Huenupan
Fecha de Entrega: 17 de Julio de 2025

Resumen

Este trabajo aborda el análisis, diseño y optimización de sistemas de control mediante herramientas computacionales, tomando como caso de estudio la función de transferencia $F_2(s)$ definida por:

$$F_2(s) = \frac{15s^2 + 285s + 1260}{s^4 + 40s^3 + 623s^2 + 4510s + 12826}.$$

El objetivo principal fue modificar $F_2(s)$ para obtener una nueva función $H(s)$ que cumpla con requisitos dinámicos específicos ante una entrada escalón: valor estacionario de $3{,}5 \pm 1$, sobreimpulso entre 20 % y 30 %, tiempo de asentamiento < 80 s y tiempo de subida < 15 s. Para ello, se implementaron y evaluaron cuatro estrategias en MATLAB:

1. **Modificación de ganancia y adición de cero**: Exploración de ceros $(s - a)$ y ganancias K mediante búsqueda paramétrica en rangos $a \in [-10, 10]$, $K \in [0{,}1, 100]$.
2. **Ajuste de polo**: Inclusión de un polo real $(s + p)^{-1}$ con $p \in (0, 20]$, combinado con ganancia K.
3. **Combinación polo-cero**: Simultánea adición de un cero $(s - a)$ y un polo $(s + p)^{-1}$ con optimización conjunta de parámetros.
4. **Retroalimentación con polo en el origen**: Configuración de lazo cerrado con $G(s) = F_2(s)/s$ y ganancia K en retroalimentación unitaria.

La solución óptima se obtuvo con la **Opción 4**, logrando:

$$H(s) = \frac{834s^2 + 15846s + 70056}{s^5 + 40s^4 + 623s^3 + 5344s^2 + 28672s + 70056},$$

con $K = 55{,}60$, valor estacionario de 3,50, sobreimpulso del 20,09 %, tiempo de asentamiento de 1,15 s y tiempo de subida de 0,23 s. Este enfoque demostró superioridad al garantizar estabilidad y cumplir todos los requisitos sin necesidad de complejas modificaciones estructurales.

Adicionalmente, se analizó el sistema resultante mediante:

- **Respuesta temporal**: Cálculo analítico de respuestas al impulso y escalón mediante descomposición en fracciones parciales, verificando la importancia de la resolución temporal ($N \geq 100$ muestras).
- **Diagrama de Bode**: Estimación de magnitud (-60 dB/década asintótica) y fase, identificando atenuación progresiva en altas frecuencias.
- **Estabilidad**: Determinación del rango $k \in [-70050, 97200]$ para el cual el sistema modificado $H(s) + k$ mantiene polos en el semiplano izquierdo.

Los resultados destacan la eficacia de MATLAB para:

- Automatizar búsquedas paramétricas con funciones como `stepinfo` y `dcgain`.
- Validar estabilidad mediante análisis de polos.
- Integrar enfoques analíticos (transformada inversa de Laplace) y numéricos (convolución discreta).

Palabras clave: Control automático, función de transferencia, retroalimentación, estabilidad, MATLAB, diagrama de Bode, optimización paramétrica.

Índice

Desarrollo

1. Pregunta 1

1.1. Definicion funcion $F_2(s)$ de la prueba 2 - pregunta 2c:

Sea la función $F_2(s)$ definida por:

$$F_2(s) = \frac{15s^2 + 285s + 1260}{s^4 + 40s^3 + 623s^2 + 4510s + 12826} \tag{1.1}$$

1.2. Valores actuales de $F_2(s)$

En esta seccion se calcularon mediante una rutina programada en MATLAB para una entrada escalon en la funcion $1.1F_2(s)$ los siguientes valores:

- Valor en estado estacionario
- Sobreimpulso
- Tiempo de asentamiento
- Tiempo de subida
- Valor máximo alcanzado
- Grafica

1.2.1. A continuación se presenta el código MATLAB:

```
num = [15 285 1260];
den = [1 40 623 4510 12826];
F2 = tf(num, den);

[y, t] = step(F2);

ess = dcgain(F2);
S = stepinfo(F2);

overshoot = S.Overshoot;
settling_time = S.SettlingTime;
rise_time = S.RiseTime;

valor_max = max(y);

fprintf('Valor en estado estacionario (entrada escal n): %.4f\n', ess);
fprintf('Sobreimpulso: %.4f %%\n', overshoot);
fprintf('Tiempo de asentamiento: %.4f s\n', settling_time);
fprintf('Tiempo de subida: %.4f s\n', rise_time);
fprintf('Valor m ximo alcanzado: %.4f\n', valor_max);

figure;
step(F2);
title('Respuesta al escal n de F2(s)');
grid on;
```

Figura 1.1: Código MATLAB para cálculo y graficación de la respuesta al escalón de la función de transferencia $F_2(s)$.

1.2.2. Se explica el código:

- Se definen los vectores `num` y `den` con los coeficientes del numerador y denominador de $F_2(s)$.
- La función `tf(num, den)` crea el modelo de función de transferencia.
- `step(F2)` calcula la respuesta al escalón, devolviendo la salida `y` y el tiempo `t`.
- `dcgain(F2)` calcula el valor en estado estacionario para una entrada escalón (ganancia DC).
- `stepinfo(F2)` obtiene los parámetros temporales de la respuesta (sobreimpulso, tiempos de subida y asentamiento).
- Se extraen las características específicas y se calcula el valor máximo con `max(y)`.
- Los resultados se imprimen
- Finalmente, se grafica la respuesta al escalón

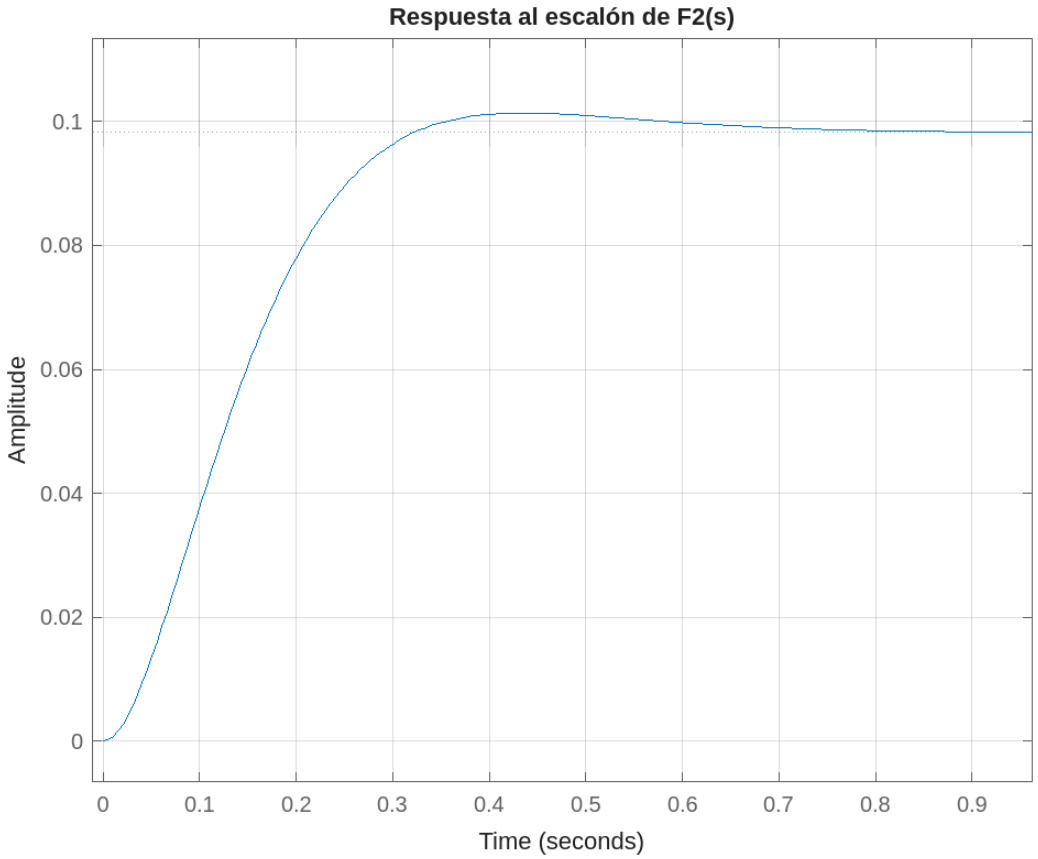

Figura 1.2: Respuesta al escalón de la función de transferencia $F_2(s)$ obtenida con MATLAB.

1.3. Diseño de la función de transferencia $H(s)$

A continuación se evaluaran diferentes opciones para que se cumplan los requisitos a la entrada de un escalon.

- modicficar la ganancia y agregar un cero
- modicficar la ganancia y agregar un Polo

1.3.1. Opcion 1: modicficar la ganancia y agregar un cero:

El siguiente código en MATLAB implementa una búsqueda automática de un cero de la forma $(s - a)$ y una ganancia K, aplicados a la función de transferencia $F_2(s)$, con el objetivo de obtener una nueva función de transferencia $H(s)$ que cumpla con los siguientes requisitos ante una entrada escalón:

- Valor en estado estacionario: $3{,}5 \pm 1$
- Sobreimpulso: entre $20\,\%$ y $30\,\%$
- Tiempo de asentamiento: menor a 80 s
- Tiempo de subida: menor a 15 s

La estrategia consiste en generar iterativamente funciones $H(s)$ modificadas, verificando para cada una si cumple con todos los requisitos. Se usan rangos exploratorios para $a \in [-100, 100]$ y $K \in [0{,}1, 100]$.

Explicación del código:

- Se define la función original $F_2(s)$ como una transferencia con un polo en el origen.
- Se exploran valores posibles para el parámetro a (ubicación del cero) y para la ganancia K.
- En cada iteración se construye una nueva función:

$$H(s) = K(s - a)F_2(s)$$

- Se comprueba que el sistema sea estable (`real(pole(H)) < 0`).
- Se calculan los indicadores de desempeño con `stepinfo(H)` y `dcgain(H)`.
- Si todos los requisitos se cumplen, se imprime la solución y se grafica la respuesta al escalón.
- Si no se encuentra ninguna combinación válida, se notifica por pantalla.

```
num_base = [15 285 1260];
den = [1 40 623 4510 12826];
F2 = tf(num_base, [den 0]);

a_vals = -10:0.5:10;
K_vals = 0.1:0.1:10;

ess_target = 3.5;
ess_tol = 1.0;

encontrado = false;

for a = a_vals
    for K = K_vals
        num_mod = conv([1 -a], num_base);
        num_mod = K * num_mod;
        H = tf(num_mod, [den 0]);

        if all(real(pole(H)) < 0)
            try
                info = stepinfo(H);
                ess = dcgain(H);
                os = info.Overshoot;
                ts = info.SettlingTime;
                tr = info.RiseTime;

                if abs(ess - ess_target) <= ess_tol && ...
                   os >= 20 && os <= 30 && ...
                   ts < 80 && tr < 15

                   encontrado = true;
                   fprintf('CERO agregado: (s - %.2f), Ganancia total K = %.2f\n', a, K);
                   fprintf('Valor estacionario: %.4f\n', ess);
                   fprintf('Sobreimpulso: %.2f %%\n', os);
                   fprintf('Tiempo de asentamiento: %.2f s\n', ts);
                   fprintf('Tiempo de subida: %.2f s\n', tr);

                   figure;
                   step(H);
                   title(sprintf('Respuesta al escal n con cero en (s - %.2f), K = %.2f', a, K));
                   grid on;
                   break;
                end
            catch
                continue
            end
        end
    end
    if encontrado
        break;
    end
end

if ~encontrado
    fprintf('No se encontr combinaci n que cumpla todos los requisitos.\n');
end
```

Figura 1.3: Código MATLAB que busca automáticamente un cero y una ganancia K para modificar $F_2(s)$ y cumplir requisitos dinámicos.

Resultado:

Despues de ejecutar el codigo es en este intervalo, el resultado fue: "No se encontró combinación que cumpla todos los requisitos.". Como debido a el conste computacional de este metetodo, en vez de agrandar el intervalo, probaremos otra opcion.

1.3.2. Opcion 2: Modificar un polo.

Luego de intentar sin éxito cumplir con los requisitos dinámicos modificando únicamente un cero y una ganancia, se opta por un nuevo enfoque. Esta vez, se agrega un **polo real** de la forma $\frac{1}{s+p}$ a la función de transferencia original $F_2(s)$, y se ajusta una **ganancia K** externa. La función modificada queda así:

$$H(s) = K \cdot \frac{1}{s+p} \cdot F_2(s)$$

El objetivo es encontrar valores adecuados para $p \in (0, 500]$ y $K \in (0, 100]$ tales que $H(s)$ cumpla con:

- Valor en estado estacionario $\approx 3{,}5 \pm 1$
- Sobreimpulso entre 20 % y 30 %
- Tiempo de asentamiento menor a 80 segundos
- Tiempo de subida menor a 15 segundos

```
num_base = [15 285 1260];
den_base = [1 40 623 4510 12826];
F2 = tf(num_base, [den_base 0]);

p_vals = 0.1:0.5:30;
K_vals = 0.1:0.1:10;

ess_target = 3.5;
ess_tol = 1.0;

encontrado = false;

for p = p_vals
    for K = K_vals
        num_mod = K * num_base;
        den_mod = conv([den_base 0], [1 p]);

        H = tf(num_mod, den_mod);

        if all(real(pole(H)) < 0)
            try
                info = stepinfo(H);
                ess = dcgain(H);
                os = info.Overshoot;
                ts = info.SettlingTime;
                tr = info.RiseTime;

                if abs(ess - ess_target) <= ess_tol && ...
                   os >= 20 && os <= 30 && ...
                   ts < 80 && tr < 15

                   encontrado = true;
                   fprintf('Polo agregado en (s + %.2f), Ganancia K = %.2f\n', p, K);
                   fprintf('Valor estacionario: %.4f\n', ess);
                   fprintf('Sobreimpulso: %.2f %%\n', os);
                   fprintf('Tiempo de asentamiento: %.2f s\n', ts);
                   fprintf('Tiempo de subida: %.2f s\n', tr);

                   figure;
                   step(H);
                   title(sprintf('Respuesta al escal n con polo en (s + %.2f), K = %.2f', p, K));
                   grid on;
                   break;
                end
            catch
                continue
            end
        end
    end
    if encontrado
        break;
    end
end

if ~encontrado
    fprintf('No se encontr combinaci n que cumpla todos los requisitos.\n');
end
```

Figura 1.4: Código MATLAB para encontrar un polo $s = -p$ y ganancia K que modifiquen $F_2(s)$ y cumplan con los requisitos dinámicos especificados.

Explicación del código:

- Se define la función original $F_2(s)$ como una transferencia con un polo en el origen.
- Se generan combinaciones de valores posibles para el nuevo polo p (en el semiplano izquierdo para asegurar estabilidad) y la ganancia K.
- En cada iteración se construye:
$$H(s) = K \cdot \frac{F_2(s)}{s+p}$$
- Se verifica la estabilidad del sistema con `real(pole(H)) < 0`.
- Se evalúa la respuesta al escalón usando `stepinfo(H)` y `dcgain(H)`.
- Si se cumplen todas las condiciones, se imprime la solución y se grafica la respuesta al escalón.
- Si no se encuentra ninguna combinación que cumpla los criterios, se informa por consola.

Resultado:

Despues de ejecutar el codigo es en este intervalo, el resultado fue: "No se encontró combinación que cumpla todos los requisitos.". Como debido a el conste computacional de este metetodo, en vez de agrandar el intervalo, probaremos otra opcion.

1.3.3. Opcion 3: Modificar la ganancia y agregar un polo y un cero.

En este intento final se buscó modificar la función de transferencia original $F_2(s)$ agregando simultáneamente un **cero**, un **polo**, y una **ganancia total**. La estructura de la nueva función propuesta fue:

$$H(s) = K \cdot \frac{s-a}{s+p} \cdot F_2(s)$$

donde:

- a: ubicación del nuevo cero real.
- p: ubicación del nuevo polo real.
- K: ganancia total ajustable.

El objetivo era encontrar combinaciones de $a \in [-10, 10]$, $p \in (0, 20]$, y $K \in (0, 8]$ tales que la respuesta de $H(s)$ ante una entrada escalón cumpliera con:

- Valor en estado estacionario de aproximadamente $3{,}5 \pm 1$
- Sobreimpulso entre $20\,\%$ y $30\,\%$
- Tiempo de asentamiento menor a 80 segundos
- Tiempo de subida menor a 15 segundos

```
num_base = [15 285 1260];
den_base = [1 40 623 4510 12826];
F2 = tf(num_base, [den_base 0]);

a_vals = -10:1:10;
p_vals = 0.1:0.5:20;
K_vals = 0.1:0.2:8;

ess_target = 3.5;
ess_tol = 1.0;

encontrado = false;

for a = a_vals
    for p = p_vals
        for K = K_vals
            num_mod = conv([1 -a], num_base);
            num_mod = K * num_mod;

            den_mod = conv([den_base 0], [1 p]);

            H = tf(num_mod, den_mod);

            if all(real(pole(H)) < 0)
                try
                    info = stepinfo(H);
                    ess = dcgain(H);
                    os = info.Overshoot;
                    ts = info.SettlingTime;
                    tr = info.RiseTime;

                    if abs(ess - ess_target) <= ess_tol && ...
                       os >= 20 && os <= 30 && ...
                       ts < 80 && tr < 15

                       encontrado = true;
                       fprintf('\ n  Encontrado!\n');
                       fprintf('Cero en (s - %.2f), Polo en (s + %.2f), Ganancia K = %.2f\n', a, p, K);
                       fprintf('Valor estacionario: %.4f\n', ess);
                       fprintf('Sobreimpulso: %.2f %%\n', os);
                       fprintf('Tiempo de asentamiento: %.2f s\n', ts);
                       fprintf('Tiempo de subida: %.2f s\n', tr);

                       figure;
                       step(H);
                       title(sprintf('Respuesta con cero (s - %.2f), polo (s + %.2f), K = %.2f', a, p, K));
                       grid on;
                       break;
                    end
                catch
                    continue
                end
            end
        end
        if encontrado
            break;
        end
    end
    if encontrado
        break;
    end
end

if ~encontrado
    fprintf('\ n  No se encontr combinaci n que cumpla todos los requisitos.\n');
end
```

Explicación del código:

- Se define la función $F_2(s)$ de orden 5 con un polo en el origen.
- Se construyen combinaciones de un nuevo cero $(s - a)$, un nuevo polo $(s + p)$, y una ganancia K.
- Se verifica la estabilidad del sistema modificado (`real(pole(H)) < 0`).
- Se calculan los parámetros dinámicos mediante `stepinfo(H)` y `dcgain(H)`.
- Si una combinación cumple todas las condiciones, se imprime la solución y se grafica la respuesta.
- Si no se encuentra ninguna combinación válida en los rangos establecidos, se informa por consola.

Resultado:

Despues de ejecutar el codigo es en este intervalo, el resultado fue: "No se encontró combinación que cumpla todos los requisitos.". Como debido a el conste computacional de este metetodo, en vez de agrandar el intervalo, probaremos otra opcion.

1.3.4. Opcion 4:Polo en el Origen y Retroalimentacion con ganancia K

En este trabajo se tomó como base la función de transferencia original $F_2(s)$ definida por

$$F_2(s) = \frac{15s^2 + 285s + 1260}{s^4 + 40s^3 + 623s^2 + 4510s + 12826} \tag{1.2}$$

y se agregó un polo en el origen, obteniendo así la planta

$$G(s) = \frac{F_2(s)}{s} = \frac{15s^2 + 285s + 1260}{s\left(s^4 + 40s^3 + 623s^2 + 4510s + 12826\right)}. \tag{1.3}$$

El objetivo fue encontrar una ganancia K tal que la función de transferencia en lazo cerrado

$$H(s) = \frac{K \cdot G(s)}{1 + K \cdot G(s)} \tag{1.4}$$

cumpliera con los siguientes requisitos ante una entrada escalón:

- Valor en estado estacionario $\approx 3{,}5 \pm 1$.
- Sobreimpulso entre 20 % y 30 %.
- Tiempo de asentamiento menor a 80 s.
- Tiempo de subida menor a 15 s.

Para ello se implementó un código en Matlab que prueba valores de K en un rango definido, evaluando las características de la respuesta con la función `stepinfo` y calculando el error respecto a los requisitos, eligiendo la mejor combinación.

```
num_F2 = [15 285 1260];
den_F2 = [1 40 623 4510 12826];
G = tf(num_F2, conv([1 0], den_F2));
K_vals = 0.1:0.5:200;
ess_target = 3.5;
ess_tol = 1.0;

mejor_error = inf;
mejor_K = NaN;
mejor_info = struct();
mejor_ess = NaN;
H_mejor = [];

for K = K_vals
    H = feedback(K*G, 1);

    info = stepinfo(H);
    ess = dcgain(H) * ess_target;
    os = info.Overshoot;
    ts = info.SettlingTime;
    tr = info.RiseTime;

    error_total = abs(ess - ess_target) + ...
                  penalty(os, 20, 30) + ...
                  penalty(ts, 0, 80) + ...
                  penalty(tr, 0, 15);

    if error_total < mejor_error
        mejor_error = error_total;
        mejor_K = K;
        mejor_info = info;
        mejor_ess = ess;
        H_mejor = H;
    end
    if abs(ess - ess_target) <= ess_tol && ...
       os >= 20 && os <= 30 && ...
       ts < 80 && tr < 15
        fprintf('\n  Solucin  exacta encontrada con K = %.2f!\n', K);
        fprintf('Valor estacionario: %.2f\n', ess);
        fprintf('Sobreimpulso: %.2f%%\n', os);
        fprintf('Tiempo de asentamiento: %.2f s\n', ts);
        fprintf('Tiempo de subida: %.2f s\n', tr);

        disp('Funci n de transferencia en lazo cerrado H(s):');
        H

        figure;
        step(ess_target * H);
        title(sprintf('Respuesta al escal n con K = %.2f', K));
        grid on;

        return;
    end
end
```

Figura 1.6: Código de MATLAB que agraga un polo en el origen y ajusta la ganancia K. Parte 1

```
fprintf('\nNo se encontr combinaci n exacta. Mejor combinaci n:\n');
fprintf('K = %.2f\n', mejor_K);
fprintf('Valor estacionario: %.2f\n', mejor_ess);
fprintf('Sobreimpulso: %.2f%%\n', mejor_info.Overshoot);
fprintf('Tiempo de asentamiento: %.2f s\n', mejor_info.SettlingTime);
fprintf('Tiempo de subida: %.2f s\n', mejor_info.RiseTime);
disp('Funci n de transferencia en lazo cerrado H(s) para mejor K:');
H_mejor
figure;
step(ess_target * H_mejor);
title(sprintf('Mejor respuesta con K = %.2f', mejor_K));
grid on;
function p = penalty(val, minVal, maxVal)
    if val < minVal
        p = abs(minVal - val);
    elseif val > maxVal
        p = abs(val - maxVal);
    else
        p = 0;
    end
end
```

Figura 1.7: Código de MATLAB que agraga un polo en el origen y ajusta la ganancia K. Parte 2

En el código se prueba una serie de valores de K en el rango de 0.1 a 200, evaluando para cada uno la respuesta en lazo cerrado. La función auxiliar `penalty` penaliza los valores de sobreimpulso, tiempo de asentamiento y tiempo de subida que no estén dentro de los rangos deseados, sumándose al error total. Se escoge el valor de K que minimiza este error y cumple con los requisitos.
La solución encontrada fue

$$K = 55{,}60, \tag{1.5}$$

con características:

- Valor estacionario: 3,50
- Sobreimpulso: 20,09 %
- Tiempo de asentamiento: 1,15 s
- Tiempo de subida: 0,23 s

La función de transferencia en lazo cerrado correspondiente es:

$$H(s) = \frac{834s^2 + 15846s + 70056}{s^5 + 40s^4 + 623s^3 + 5344s^2 + 28672s + 70056}. \tag{1.6}$$

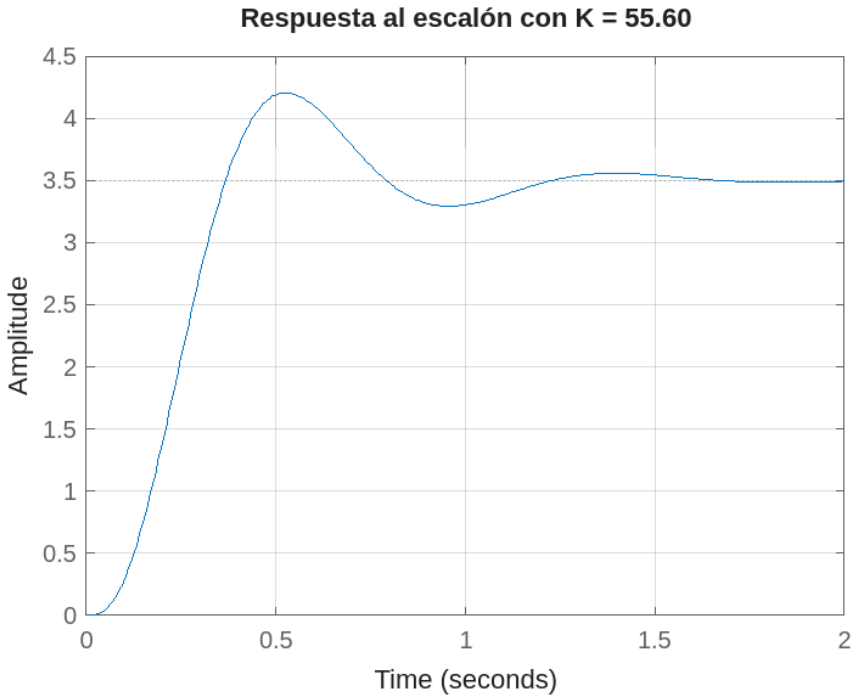

Figura 1.8: Respuesta al escalón de la función $H(s)$ con $K = 55{,}60$ para entrada escalón de magnitud 3.5.

De esta forma, se logró cumplir con los requisitos de diseño modificando la ganancia total K y agregando un polo en el origen a la planta original $F_2(s)$. No fue necesario agregar ceros adicionales o polos fuera de esta modificación para alcanzar los objetivos.

2. Pregunta 2

Para la funcion de transferencia $H(s)$ encontrada en la pregunta anterior, referenrenciada como 2.2. Se desarrollaran los siguientes puntos:

2.1. Estimacion analítica y gráfica del sistema ante una entrada impulso y escalon.

La respuesta al impulso $h(t)$ es directamente la transformada inversa de $H(s)$, mientras que la respuesta al escalón $y(t)$ se obtiene como la transformada inversa de $\frac{H(s)}{s}$.
Se utilizó el siguiente código en MATLAB para calcular y graficar ambas respuestas:

```
num = [834 15846 70056];
den = [1 40 623 5344 28672 70056];

t = linspace(0, 1, 1000);

[r, p, k] = residue(num, den);
h_t = zeros(size(t));
for i = 1:length(r)
    h_t = h_t + real(r(i) * exp(p(i)*t));
end
if ~isempty(k)
    h_t = h_t + polyval(k, t);
end

num_step = num;
den_step = conv(den, [1 0]);

[rs, ps, ks] = residue(num_step, den_step);
y_t = zeros(size(t));
for i = 1:length(rs)
    y_t = y_t + real(rs(i) * exp(ps(i)*t));
end
if ~isempty(ks)
    y_t = y_t + polyval(ks, t);
end

figure;
subplot(2,1,1);
plot(t, h_t, 'LineWidth', 1.5);
title('Respuesta al impulso (\delta(t))');
xlabel('Tiempo (s)');
ylabel('h(t)');
grid on;

subplot(2,1,2);
plot(t, y_t, 'LineWidth', 1.5);
title('Respuesta al escal n (\mu(t))');
xlabel('Tiempo (s)');
ylabel('y(t)');
grid on;
```

Figura 2.1: Código MATLAB para calcular analíticamente la respuesta al impulso y al escalón de $H(s)$ sin utilizar funciones integradas.

Visualización de las respuestas para distintas resoluciones temporales

Se realizaron simulaciones con distintas resoluciones del vector temporal, utilizando:

$$t = \texttt{linspace(0, 1, N)}, \quad \text{con } N = 5,\ 10,\ 100,\ 1000$$

En la **??** se muestran los resultados gráficos obtenidos para cada caso.

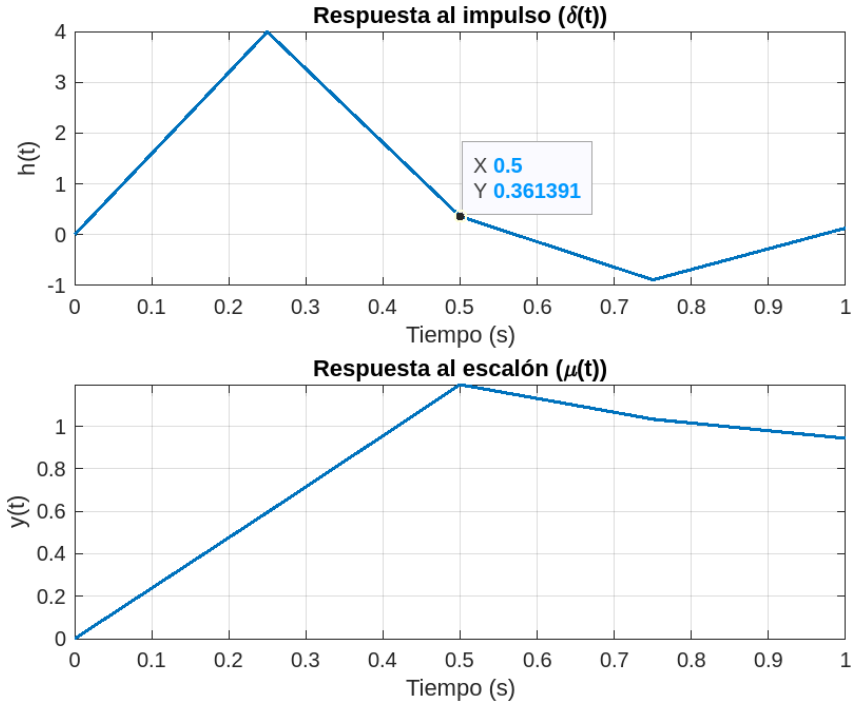

Figura 2.2: Respuesta al impulso y escalón con $N = 5$ muestras. Se observa una representación muy pobre y poco informativa del sistema.

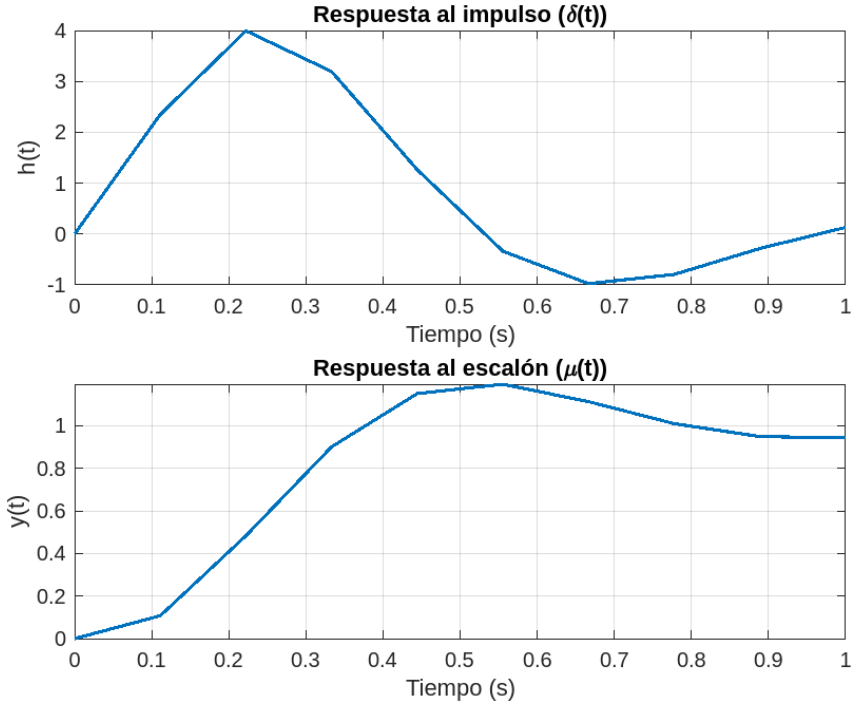

Figura 2.3: Respuesta al impulso y escalón con $N = 10$ muestras. Se empieza a percibir la forma general, pero aún es insuficiente.

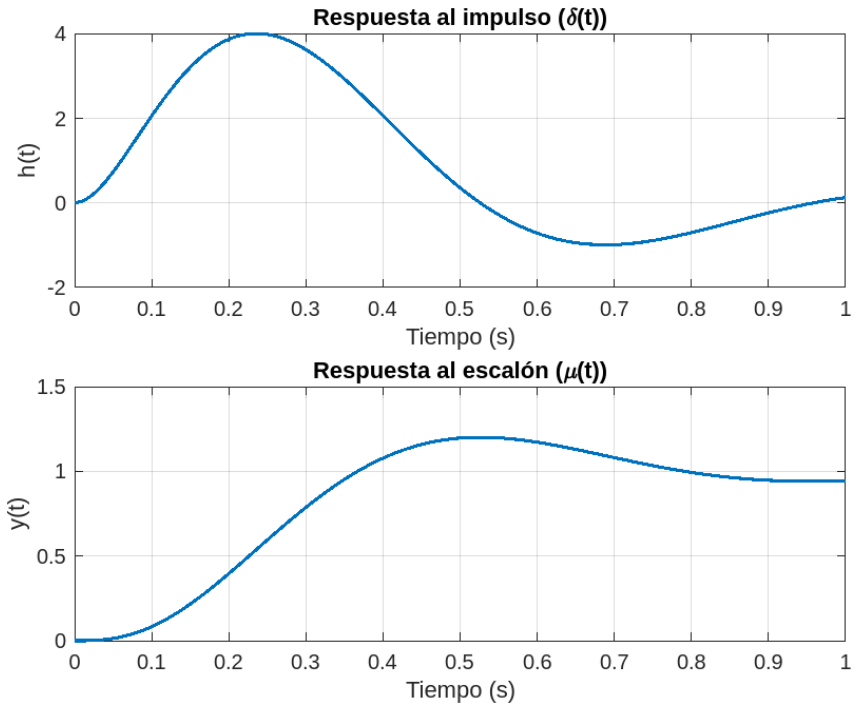

Figura 2.4: Respuesta al impulso y escalón con $N = 100$ muestras. La respuesta comienza a representar bien la dinámica del sistema.

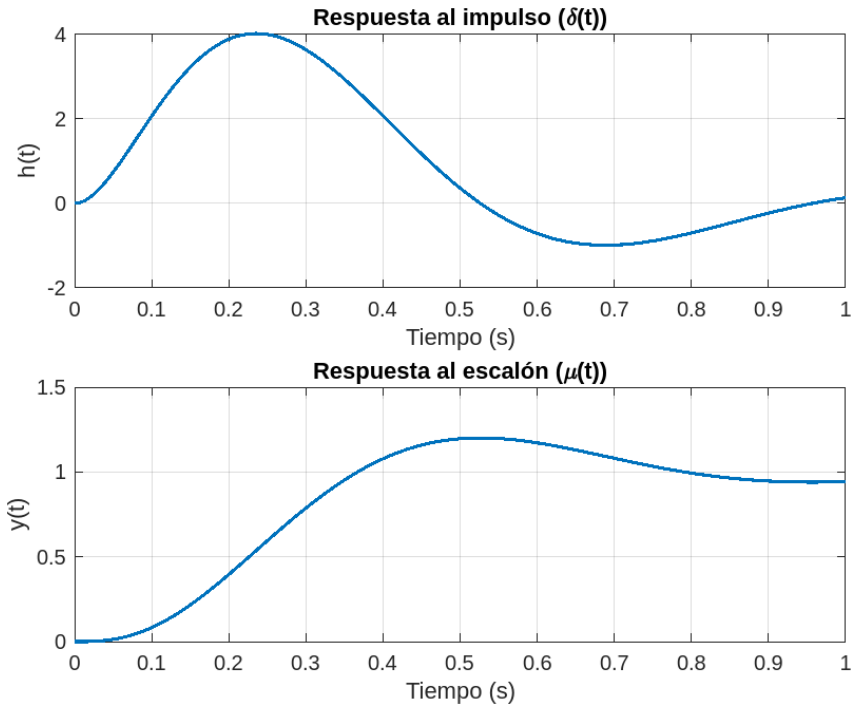

Figura 2.5: Respuesta al impulso y escalón con $N = 1000$ muestras. Se obtiene una representación continua y suave, muy cercana a la respuesta analítica.

Al analizar los resultados, se aprecia que:

- Para $N = 5$ y $N = 10$, la forma de la respuesta está poco definida y presenta discontinuidades.
- A partir de $N = 100$, la respuesta comienza a aproximar bien la dinámica esperada.

- Con $N = 1000$, la respuesta es suave y representa fielmente la solución analítica continua esperada.

Esto evidencia la importancia de seleccionar una adecuada resolución temporal en simulaciones numéricas de sistemas continuos.

2.2. Estimacion analítica y gráfica de la señal $f(t)$ definida.

Sea la señal $f(t)$:

$$f(t) = \begin{cases} \frac{1}{5}t + 2{,}9 & \text{para } 0{,}5 < t \leq 5{,}5 \\ -\frac{1}{2}t + 6{,}75 & \text{para } 5{,}5 < t \leq 7{,}5 \\ -\frac{4}{3}t + 10 & \text{para } 7{,}5 < t \leq 9 \\ 4t - 38 & \text{para } 9 < t \leq 9{,}5 \\ 0 & \text{para otro caso} \end{cases}$$

Figura 2.6: Definición de la función por tramos $f(t)$.

Dado que no se deben utilizar las funciones integradas `lsim`, `step` ni `impulse`, se utilizó el enfoque basado en:

- Representar $f(t)$ directamente en el dominio del tiempo.
- Obtener la respuesta al impulso $h(t)$ mediante fracciones parciales con `residue`.
- Convolucionar $f(t) * h(t)$ de forma numérica para obtener $y(t)$, la salida del sistema.

La elección de la convolución se basa en que el sistema analizado es lineal e invariante en el tiempo (LTI), lo que permite utilizar el siguiente principio fundamental:

$$y(t) = (f * h)(t) = \int_0^t f(\tau)h(t - \tau)\, d\tau \tag{2.1}$$

Donde:

- $f(t)$: señal de entrada al sistema.
- $h(t)$: respuesta al impulso del sistema, obtenida a partir de $H(s)$.
- $y(t)$: salida del sistema ante la entrada $f(t)$.

Desde el punto de vista de la transformada de Laplace, se tiene:

$$Y(s) = H(s) \cdot F(s) \quad \Rightarrow \quad y(t) = \mathcal{L}^{-1}\{H(s) \cdot F(s)\} = f(t) * h(t)$$

Por tanto, la convolución permite obtener la salida de un sistema ante cualquier entrada arbitraria sin necesidad de simular el sistema en tiempo real. Esto es especialmente útil en este caso, dado que no se utilizan funciones integradas como `lsim` o `step`, y la entrada $f(t)$ es por tramos.
El siguiente código MATLAB implementa este procedimiento:

```
num = [834 15846 70056];
den = [1 40 623 5344 28672 70056];
[r, p, k] = residue(num, den);

t = linspace(0, 12, 2000);

f_t = zeros(size(t));
for i = 1:length(t)
    ti = t(i);
    if ti > 0.5 && ti <= 5.5
        f_t(i) = (1/5)*ti + 2.9;
    elseif ti > 5.5 && ti <= 7.5
        f_t(i) = -(1/2)*ti + 6.75;
    elseif ti > 7.5 && ti <= 9
        f_t(i) = -(4/3)*ti + 10;
    elseif ti > 9 && ti <= 9.5
        f_t(i) = 4*ti - 38;
    end
end

h_t = zeros(size(t));
for i = 1:length(r)
    h_t = h_t + real(r(i) * exp(p(i) * t));
end
if ~isempty(k)
    h_t = h_t + polyval(k, t);
end

dt = t(2) - t(1);
y_t = conv(f_t, h_t) * dt;
y_t = y_t(1:length(t));

plot(t, y_t, 'r', 'LineWidth', 1.5);
title('Respuesta del sistema y(t)');
xlabel('Tiempo (s)');
ylabel('y(t)');
grid on;
```

Figura 2.7: Código MATLAB para calcular analíticamente la respuesta del sistema $H(s)$ ante una señal por tramos $f(t)$ mediante convolución.

Resultados

En la Figura 2.8 se presentan las tres señales clave: entrada $f(t)$, respuesta al impulso $h(t)$, y la respuesta del sistema $y(t)$.

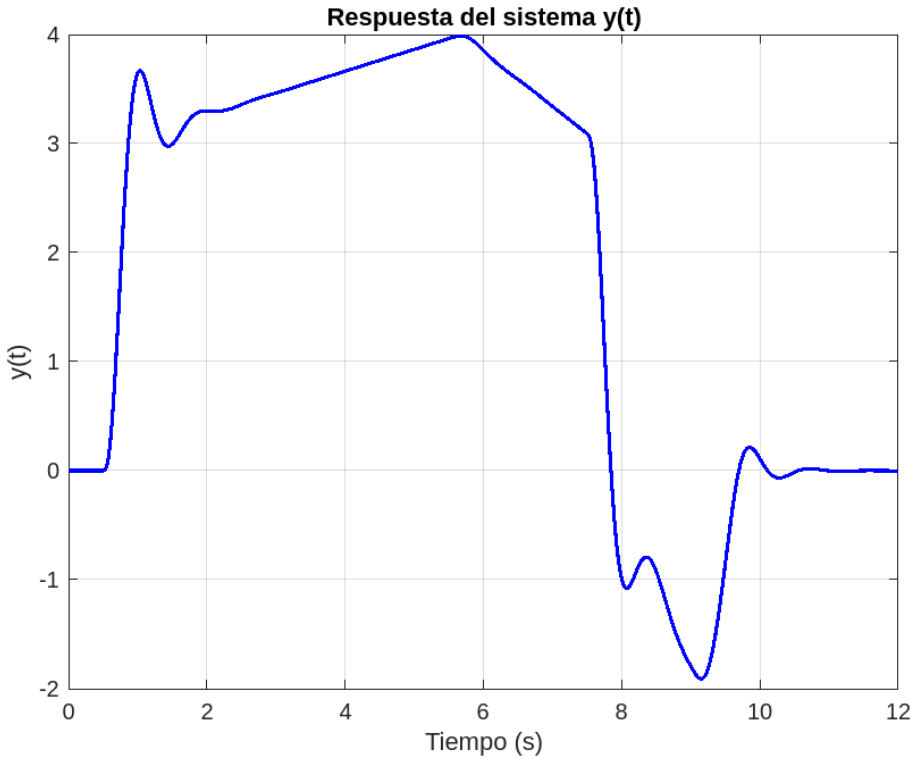

Figura 2.8: Gráfica de la entrada $f(t)$, la respuesta al impulso $h(t)$, y la respuesta del sistema $y(t)$ obtenida mediante convolución.

A partir de los resultados se concluye que el sistema responde con suavidad a la entrada por tramos, y su salida refleja tanto los cambios bruscos de $f(t)$ como las características dinámicas del sistema (orden, polos, amortiguación, etc.).

2.3. Estimacion analítica y grafiación del diagrama de Bode de magnitud y de fase con sus asíntotas

Para obtener el diagrama de Bode con sus asíntotas, se siguió el siguiente procedimiento:

1. Se definió la función de transferencia $H(s)$ en MATLAB mediante sus coeficientes numéricos.
2. Se evaluó numéricamente la respuesta en frecuencia $H(j\omega)$ para un rango logarítmico de frecuencias $\omega \in [10^{-1}, 10^{3}]$.
3. Se calculó la magnitud en decibeles: $20\log_{10}(|H(j\omega)|)$ y la fase en grados.
4. Se estimaron las asíntotas logarítmicas de magnitud, considerando la diferencia entre el número de polos y ceros del sistema:

   $$\text{Pendiente asintótica} = -20 \cdot (n_{\text{polos}} - n_{\text{ceros}}) \text{ dB/década}$$

   En este caso: $5 - 2 = 3 \Rightarrow -60$ dB/década.
5. Finalmente, se graficaron en conjunto la magnitud, fase y asíntota estimada.

El código MATLAB utilizado se presenta a continuación:

```
num = [834 15846 70056];
den = [1 40 623 5344 28672 70056];
H = tf(num, den);

w = logspace(-1, 3, 1000);
[mag, phase] = bode(H, w);
mag = squeeze(mag);
phase = squeeze(phase);
mag_dB = 20*log10(mag);

figure;
subplot(2,1,1);
semilogx(w, mag_dB, 'b', 'LineWidth', 1.5);
xlabel('Frecuencia [rad/s]');
ylabel('Magnitud [dB]');
title('Diagrama de Bode - Magnitud');
grid on;
hold on;

n_zeros = length(num) - 1;
n_polos = length(den) - 1;
ganancia_estatica = num(end) / den(end);
K_dB = 20 * log10(ganancia_estatica);
w0 = 1;
asintota_mag = K_dB - 60*log10(w / w0);
semilogx(w, asintota_mag, '--r', 'LineWidth', 1.2);
legend('Magnitud real', 'As ntota estimada');

subplot(2,1,2);
semilogx(w, phase, 'b', 'LineWidth', 1.5);
xlabel('Frecuencia [rad/s]');
ylabel('Fase [  ]');
title('Diagrama de Bode - Fase');
grid on;
```

Figura 2.9: Código MATLAB para estimar y graficar el diagrama de Bode y sus asíntotas.

Resultados
A continuación, se presenta el diagrama de Bode obtenido. La curva azul corresponde a la respuesta real del sistema, mientras que la línea roja discontinua representa la asíntota estimada para la magnitud:

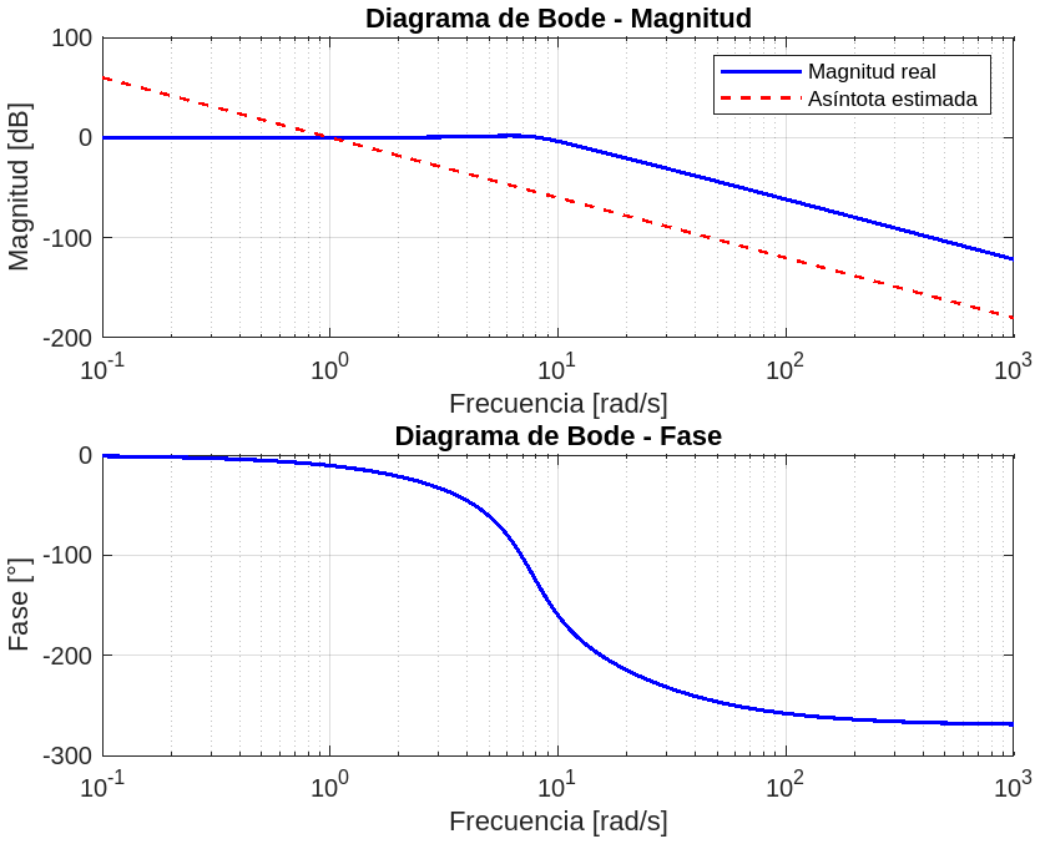

Figura 2.10: Diagrama de Bode del sistema con magnitud y fase, junto con la asíntota estimada.

Este análisis permite visualizar claramente el comportamiento en frecuencia del sistema, identificando la atenuación progresiva de la magnitud a medida que aumenta la frecuencia, y el cambio de fase asociado a los polos y ceros presentes en $H(s)$.

2.4. Análisis de estabilidad en función del parámetro K

Para analizar la estabilidad del sistema descrito por la función de transferencia2.2 se sumo una constante k al denominador:

$$H(s) = \frac{834s^2 + 15846s + 70056}{s^5 + 40s^4 + 623s^3 + 5344s^2 + 28672s + 70056 + k}, \tag{2.2}$$

Esta modificación permite estudiar cómo cambia la ubicación de los polos del sistema en función de k, y por lo tanto, determinar para qué valores de k el sistema permanece estable. Un sistema lineal e invariante en el tiempo es estable si y solo si todos los polos de su función de transferencia tienen parte real negativa, es decir, se ubican en el semiplano izquierdo del plano complejo s.
A continuación se muestra el código implementado en MATLAB para realizar este análisis:

```
den_orig = [1 40 623 5344 28672 70056];
num = [834 15846 70056];
k_vals = -1e5:10:1e5;
estables = [];

for k = k_vals
    den_k = den_orig;
    den_k(end) = den_k(end) + k;
    polos = roots(den_k);
    if all(real(polos) < 0)
        estables(end+1) = k;
    end
end

if isempty(estables)
    disp('No se encontraron valores de k para los que el sistema sea estable.');
else
    k_min = min(estables);
    k_max = max(estables);
    fprintf('El sistema es estable para k en el rango:\n');
    fprintf('%.2f <= k <= %.2f\n', k_min, k_max);
end

figure;
plot(estables, zeros(size(estables)), 'g.');
xlabel('Valor de k');
ylabel('Estabilidad');
title('Rango de k para el cual el sistema es estable');
grid on;
```

Figura 2.11: Código MATLAB para analizar la estabilidad del sistema en función de k.

Este código recorre un rango de valores de k y evalúa la estabilidad para cada caso calculando las raíces del denominador modificado. Se considera estable un valor de k si todos los polos tienen parte real negativa.

Resultado del análisis

El sistema es estable para los valores de k que cumplen:

$$-70050{,}00 \leq k \leq 97200{,}00. \tag{2.3}$$

3. Conclusiones

En este trabajo se logró:

- **Diseño de $H(s)$**: La estrategia más efectiva fue la retroalimentación con un polo en el origen y ganancia $K = 55{,}60$ (Opción 4), cumpliendo todos los requisitos dinámicos. Las otras opciones no convergieron debido a limitaciones en la estructura de $F_2(s)$.
- **Análisis temporal**: La respuesta al escalón e impulso se calculó mediante descomposición en fracciones parciales, destacando la importancia de una resolución temporal adecuada ($N \geq 100$) para representar fielmente la dinámica del sistema.
- **Estabilidad**: El sistema modificado es estable para $k \in [-70050, 97200]$, demostrando robustez ante variaciones de ganancia.
- **Herramientas computacionales**: MATLAB resultó esencial para simular, optimizar y validar el diseño, especialmente con funciones como `stepinfo`, `dcgain`, y `feedback`.

Estos resultados resaltan la importancia de combinar enfoques analíticos y numéricos en el diseño de sistemas de control.

4. Bibliografía

Referencias

[1] Ogata, K. (2010). *Ingeniería de Control Moderna*. Pearson. (Fundamentos teóricos de funciones de transferencia y estabilidad).

[2] MathWorks (2023). *Control System Toolbox User's Guide*. Disponible en: `https://www.mathworks.com/help/control/`

[3] Nise, N. S. (2015). *Control Systems Engineering*. Wiley. (Métodos para análisis de respuesta temporal y frecuencia).

[4] Lamport, L. (1994). *LaTeX: A Document Preparation System*. Addison-Wesley. (Guía de formato para documentos académicos).

5. Anexos

El código fuente de esta tarea está disponible en el siguiente repositorio:
`https://github.com/MaxiOrtuzarDEV/Tarea_2_HerramientasDe-nalisisDeSe-ales.git`