Práctica de Laboratorio 3: Diseño de controladores por el método de Diagramas de Bode.

Elías Álvarez
Carrera de Ing. Electrónica
Universidad Católica Nuestra Señora de la Asunción
Asunción, Paraguay
Email: elias.alvarez@universidadcatolica.edu.py

Tania Romero
Carrera de Ing. Electrónica
Universidad Católica Nuestra Señora de la Asunción
Asunción, Paraguay
Email: tania.romero@universidadcatolica.edu.py

Docente: Lic. Montserrat González
Facultad de Ingeniería
Universidad Católica Nuestra Señora de la Asunción
Asunción, Paraguay

Docente: PhD. Enrique Vargas
Facultad de Ingeniería
Universidad Católica Nuestra Señora de la Asunción
Asunción, Paraguay

I. Indicadores

- Estabilidad en lazo cerrado: polos dentro del círculo unitario en z.
- Tiempo de subida: con $\zeta = 0.7$ y 8 muestras en t_r.
- Error en estado estacionario (ESS): debe ser igual a cero.
- Comparación Matlab vs. experimento.

II. Introducción

En este laboratorio se aplica el método del lugar de raíces para el diseño de un controlador digital que modifique la dinámica de una planta previamente discretizada mediante retención de orden cero. A partir de las especificaciones de desempeño (tiempo de subida, factor de amortiguamiento y error en estado estacionario), se determinan los polos deseados en el plano-z y se ajusta el compensador para que dichos polos pertenezcan al lugar de raíces del sistema. De este modo, se logra un diseño sistemático que permite cumplir con los requisitos de estabilidad y respuesta transitoria, comparando los resultados obtenidos en Matlab con la implementación práctica en PSoC.

III. Objetivos

- Diseñar un controlador que modifique la dinámica de la planta para satisfacer condiciones específicas de la respuesta transitoria del sistema de control en lazo cerrado.
- El sistema regulado debe ser estable.
- El error en estado estacionario (ESS) debe ser igual a cero.
- Observar y analizar los efectos del controlador en el comportamiento del sistema.
- Considerar diferentes métodos para el ajuste de los parámetros del controlador y analizar los resultados.
- Diseñar el sistema de control en Matlab e implementar la ecuación en diferencias en PSoC.

IV. Materiales

- PC con Matlab.
- Planta analógica.
- Sistema de adquisición en PSoC.

V. Teoría

Véase K. Ogata, *Sistemas de Control en Tiempo Discreto*, págs. 204–225.

VI. Desarrollo

VI-A. Modelado del Sistema

VI-A1. Obtención de la función de transferencia en lazo abierto: La planta puede interpretarse como la conexión en cascada de dos filtros activos de primer orden. Cada uno posee la misma topología: un amplificador operacional en configuración inversora cuya impedancia de realimentación está compuesta por una resistencia en paralelo con un capacitor.

VI-A1a. Impedancia de realimentación.: Para el paralelo $R_f \parallel C_f$, se obtiene:

$$Z_f \;=\; R_f \parallel \frac{1}{sC_f} \;=\; \frac{R_f}{1 + sR_fC_f} \tag{1}$$

El sistema trabaja sobre una tensión de referencia en continua $V_{\text{cc}}/2 = 2.5$ V. En los cálculos posteriores se toma dicho valor como punto de referencia.

VI-A1b. Ganancia de una etapa (entrada inversora).: Con $V_{\text{ref}} = 0$, se cumple el cortocircuito virtual ($V_p = V_n = 0$). Aplicando KCL en el nodo inversor:

$$\frac{V_i}{R_i} \;=\; \frac{-V_o}{Z_f} \quad \Rightarrow \quad \frac{V_o}{V_i} \;=\; -\,\frac{Z_f}{R_i}$$

y reemplazando (1):

$$\left.\frac{V_o}{V_i}\right|_{V_{\text{ref}}=0} = -\,\frac{R_f}{R_i}\,\frac{1}{1 + sR_fC_f} \tag{2}$$

VI-A1c. Encadenamiento de etapas.: Como ambas etapas AO1 y AO2 responden a la forma (2), la ganancia total en lazo abierto resulta del producto de sus transferencias:

$$G_{\text{ol}}(s) = \Big(-\frac{Z_{f1}(s)}{R_{i1}} \Big) \Big(-\frac{Z_{f2}(s)}{R_{i2}} \Big)$$

VI-A1d. Implementación en Matlab.: El siguiente código genera cada etapa de primer orden, calcula la función en lazo abierto y extrae información temporal de la respuesta al escalón:

VI-A1e. Resultados numéricos.: A partir de la respuesta de la figura **??** se obtuvieron:

- **Tiempo de subida:** $t_r \approx 0.0332$ s (33.2 ms)
- **Tiempo de establecimiento (2 %):** $t_s \approx 0.0603$ s (60.3 ms)
- **Frecuencia natural estimada:** $\omega_n \approx 54.2$ rad/s

VI-B. Discretización de la Planta

2.1. Elegir el tiempo de muestreo T_s para que existan 8 muestras dentro del tiempo de subida. Para este caso se considera un factor de amortiguamiento $\zeta = 0.7$.

La planta a discretizar está dada por:

$$G(s) = \frac{K}{(s+a)(s+b)} \tag{3}$$

Este sistema corresponde a una planta de segundo orden, representada como el producto de dos polos reales.
Para discretizar, partimos del método de la *transformación de la respuesta al impulso*. La relación general es:

$$G(z) = (1 - z^{-1})\, \mathcal{Z}\left[\mathcal{L}^{-1}\left(\frac{G(s)}{s} \right)^{*} \right] \tag{4}$$

Es decir, primero se obtiene la respuesta al escalón unitario en el tiempo continuo, luego se transforma con $\mathcal{Z}$ y finalmente se multiplica por $(1 - z^{-1})$ para obtener la función de transferencia discreta.

Desarrollo de fracciones parciales: Se descompone $\frac{G(s)}{s}$ en términos más simples para poder aplicar transformadas inversas de Laplace:

$$\frac{G(s)}{s} = \underbrace{\frac{K}{ab}}_{\delta} \cdot \frac{1}{s} + \underbrace{\frac{K}{b^2 - ab}}_{\beta} \cdot \frac{1}{s+b} + \underbrace{\frac{K}{a^2 - ab}}_{\alpha} \cdot \frac{1}{s+a} \tag{5}$$

Aquí aparecen tres coeficientes, que llamamos δ, β y α, cada uno asociado a un término simple de la expansión.

Respuesta en el tiempo continuo: Aplicando la transformada de Laplace inversa, cada término corresponde a una exponencial decreciente o a una constante:

$$\mathcal{L}^{-1}\left\{ \frac{G(s)}{s} \right\} = \delta + \beta e^{-bTk} + \alpha e^{-aTk}, \quad k \geq 0 \tag{6}$$

Este resultado representa la respuesta al escalón unitario en tiempo discreto, evaluada en múltiplos de T.

Transformada Z: Al transformar esta secuencia al dominio-z, se obtiene:

$$\mathcal{Z}\left[\frac{G(s)}{s} \right] = \delta \frac{z}{z-1} + \beta \frac{z}{z - e^{-bT}} + \alpha \frac{z}{z - e^{-aT}} \tag{7}$$

Cada término se convierte en una fracción simple con un polo en $z = 1$, $z = e^{-bT}$ y $z = e^{-aT}$ respectivamente.

Función de transferencia discreta: Finalmente, al multiplicar por $(1 - z^{-1})$, se obtiene la expresión general de la planta discretizada, G(z), vista enteriormente en la ecuación 4.

Cálculo de coeficientes: Con los valores $a = \frac{1}{81\,000 \cdot 200 \cdot 10^{-9}}$ y $b = \frac{1}{15 \cdot 10^{3} \cdot 100 \cdot 10^{-9}}$, y considerando $K = 1$ y $T = 1.2452$, se obtienen los siguientes coeficientes:

$$\delta = \frac{K}{ab} \approx 24.3 \times 10^{6}, \qquad \beta = \frac{K}{b^2 - ab} \approx -2.4 \times 10^{6},$$

$$\alpha = \frac{K}{a^2 - ab} \approx -26.78 \times 10^{6}$$

Planta discretizada: Reemplazando los valores en la expresión final, se obtiene aproximadamente:

$$G(z) \approx \frac{0.022(z + 0.74)}{(z - 0.93)(z - 0.44)} \tag{8}$$

2.2. Comparar este resultado con el obtenido en Matlab utilizando el comando `c2d`.

$$H(z) = \frac{0.021596\,(z + 0.7419)}{(z - 0.9333)(z - 0.436)}$$

Se puede observar que los valores son muy aproximados.

VI-C. Modificación de la Dinámica

Una vez digitalizada la planta, es posible identificar la ubicación de polos y ceros en el plano-z (véase la Fig. 1).

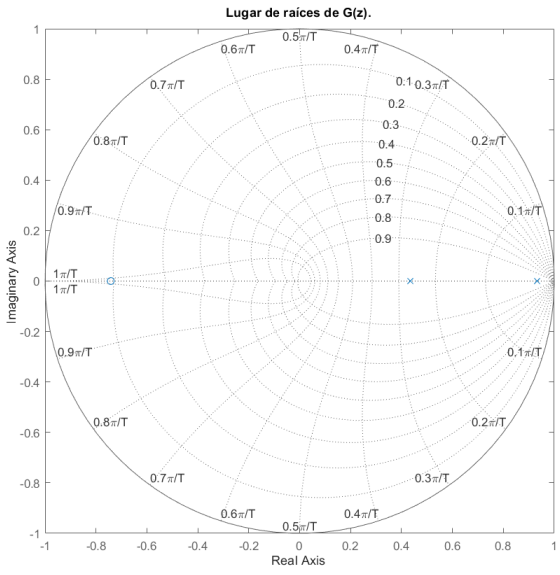


Figura 1: Lugar de las raíces de la planta digitalizada con el tiempo de muestreo T calculado.

De acuerdo con las especificaciones solicitadas, se debe determinar un punto en el plano-z que satisfaga las condiciones de desempeño requeridas.

Recordando que

$$z = e^{sT}, \quad \text{con } s = \sigma + j\omega_d,$$

donde $\sigma = \zeta \cdot \omega_n$ y $\omega_d = \omega_n\sqrt{1-\zeta^2}$, se obtiene un punto candidato p_{target}, cuyas coordenadas en el plano-z son

$$x = 0.843273, \qquad y = 0.136677.$$

Este punto representa la ubicación de los polos deseados que garantizarían la aproximación de segundo orden y, por ende, las características dinámicas solicitadas.

Para verificar si el punto pertenece al lugar de las raíces de la planta, se aplica la *condición de fase*, la cual establece que la suma de ángulos debe satisfacer la Ec. (9):

$$\begin{aligned} \angle(p_{\text{target}} - z_0) - \angle(p_{\text{target}} - p_0) \\ - \angle(p_{\text{target}} - p_1) \quad &= \phi \\ &\approx -136.42^\circ \end{aligned} \tag{9}$$

De este análisis se deduce que es necesario introducir un compensador que aporte un desfase adicional de -43.042° para que el punto deseado forme parte del lugar de raíces y el sistema cumpla con las especificaciones de diseño.

3.1. Diseñar compensador para $\zeta = 0.7$: Para el primer compensador, a partir del análisis del *ángulo faltante* ϕ para cumplir con la condición de fase (Ec. (9)), la alternativa más sencilla consiste en diseñar un filtro *lag* en cascada de primer orden.

Una opción es cancelar uno de los polos originales de la planta mediante la ubicación de un cero en la misma posición, y colocar a continuación un nuevo polo de manera que el aporte de fase conjunto de estos dos compense el desfase requerido. De esta forma, el punto seleccionado pasará a pertenecer al lugar de las raíces.

En este caso, se procede a cancelar el polo más lento del sistema. Para ello, se analiza el triángulo formado por el cero, el nuevo polo y el punto deseado en el plano-z, representado en la Figura 2.

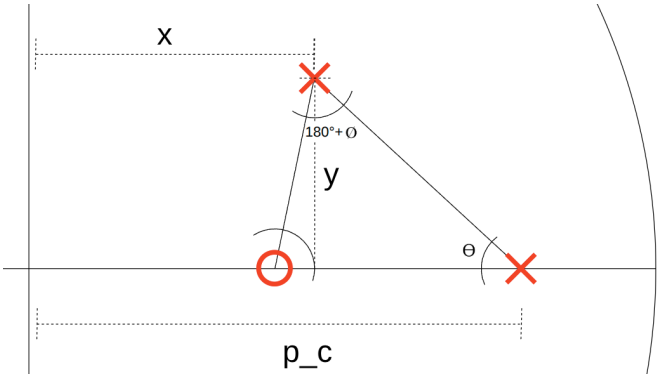


Figura 2: Análisis geométrico para el diseño del primer compensador (no es a escala).

Ecuacionando, por la ley de senos se obtienen las siguientes 2 relaciones:

$$\theta = \arcsin\left(\frac{y}{\sqrt{(x-z_c)^2+y^2}}\right) \tag{10}$$

$$\frac{p_c - z_c}{\sin(180^\circ + \phi)} = \frac{\sqrt{(x-z_c)^2+y^2}}{\sin(-\phi+\theta)} \tag{11}$$

Usando las Ec. (10) y (11), se concluye que:

$$p_c = \frac{\sin(180^\circ+\phi)\sqrt{(x-z_c)^2+y^2}}{\sin\left(-\arcsin\left(\frac{y}{\sqrt{(x-z_c)^2+y^2}}\right)-\phi\right)} + z_c \tag{12}$$

Con la Ec. (12), se obtiene un filtro *lag* que asegura que el punto deseado pertenezca al lugar de las raíces. Restará determinar el valor de la ganancia K que garantice que los polos en lazo cerrado se ubiquen efectivamente en dicha posición.

Esto puede realizarse aplicando el *criterio de magnitud*, expresado en la Ec. (13):

$$K = \frac{\prod_i \sqrt{(p_i - x)^2 + y^2}}{\prod_j \sqrt{(z_j - x)^2 + y^2}} \tag{13}$$

No obstante, también es posible emplear directamente la función `rlocus` de MATLAB, lo cual facilita el cálculo.

Cuadro I: Parámetros del Compensador 1

z_0	0.436
p_0	0.769221
K	0.743

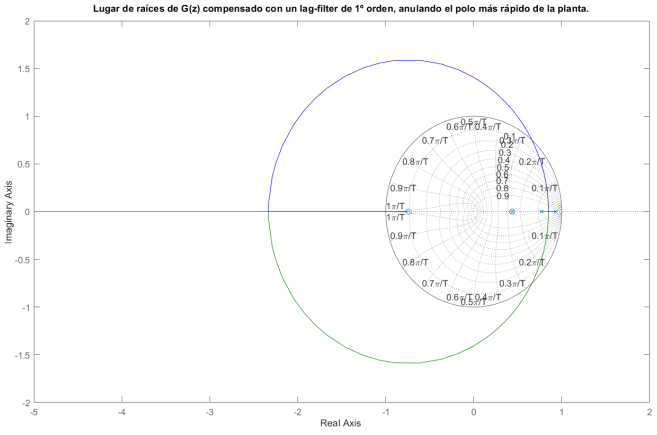


Figura 3: Lugar de las raíces de la planta digitalizada y compensada con un filtro lag de primer orden.

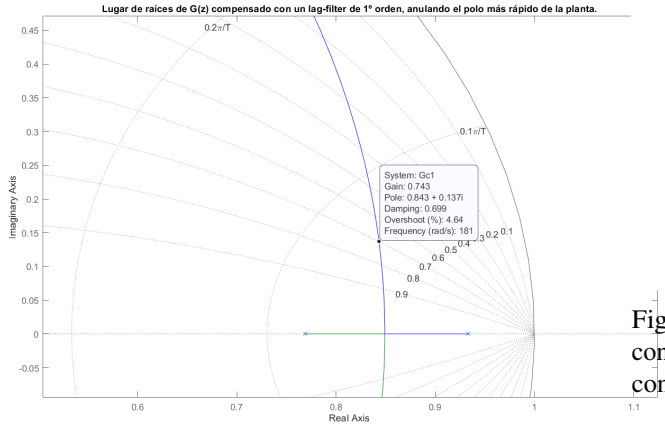


Figura 4: Lugar de las raíces de la planta digitalizada y compensada con un filtro lag de primer orden (detalle ampliado).

Una vez identificado el compensador y la ganancia, se procede a cerrar el lazo y analizar la respuesta en el tiempo. La Fig. 5 muestra la respuesta al escalón con el primer compensador diseñado.

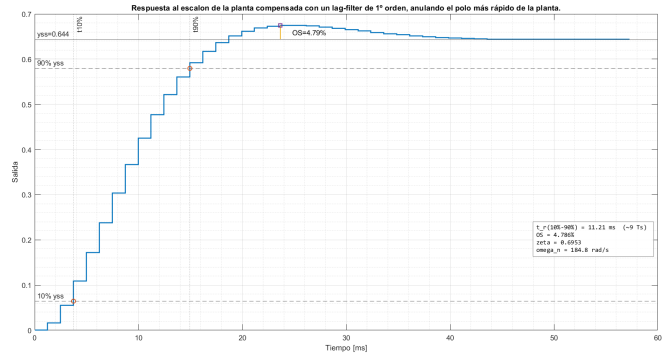


Figura 5: Respuesta al escalón del sistema en lazo cerrado con el primer compensador diseñado.

Se observa que se cumplen las condiciones de desempeño solicitadas; sin embargo, aparece un error en estado estacionario considerable. Esto se debe a que el sistema en lazo abierto $K \cdot C(z) \cdot G_d(z)$ es de tipo 0, por lo cual no posee la capacidad de seguir correctamente a una entrada escalón.
Asimismo, se evalúa el esfuerzo de control aplicado (Fig. 6). Si bien resulta implementable, se aprecia que presenta una magnitud inicial elevada con picos pronunciados.

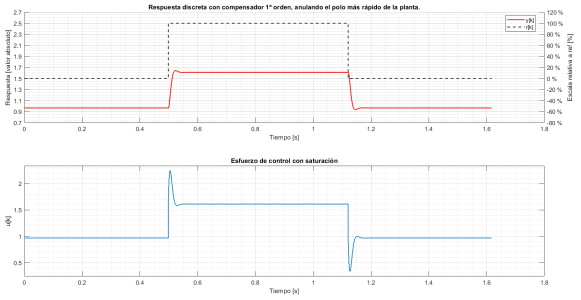


Figura 6: Respuesta al escalón del sistema en lazo cerrado con el primer compensador diseñado, junto con el esfuerzo de control aplicado por el compensador.

3.2. Diseñar compensador para $\zeta = 0.7$ y ESS $= 0$: Para el segundo compensador solicitado, con el fin de cumplir la condición de que el error en estado estacionario sea cero, necesitamos que la función de transferencia de lazo abierto sea de tipo 1. Sin embargo, la planta original es de tipo 0, lo cual nos obliga a colocar un polo en $z = 1$. Como primer intento, buscamos un cero que, junto con el polo en $z = 1$, compense el ángulo faltante de manera similar al diseño del primer compensador. Para ello desarrollamos la Ec. (14):

$$z_c = p_c \; - \; \frac{\sin(\pi + \phi)\,\sqrt{(x - p_c)^2 + y^2}}{\sin\!\left(-\arcsin\!\left(\dfrac{y}{\sqrt{(x-p_c)^2+y^2}}\right) \; - \; \phi\right)}. \quad (14)$$

Inicialmente este enfoque parece prometedor: podemos calcularlo (valores obtenidos) y, utilizando MATLAB, obtener la ganancia K correspondiente (véase Fig. 7).

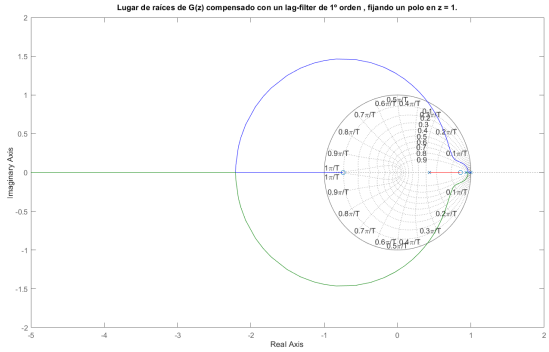


Figura 7: Lugar de raíces de la planta compensada con un filtro *lag* de primer orden y polo en $z = 1$.

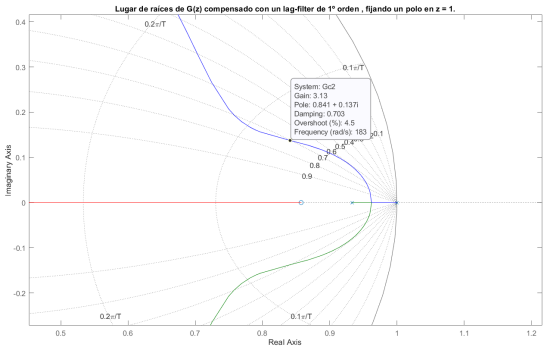


Figura 8: Lugar de raíces de la planta compensada con un filtro *lag* de primer orden y polo en $z = 1$ (detalle ampliado).

No obstante, al observar la respuesta al escalón nos encontramos con una decepción: se obtiene un sobrepico del $20\,\%$, equivalente a un $\zeta \approx 0.4$, lo que no satisface las condiciones deseadas.

Cuadro II: Parámetros del Compensador 2

z_0	0.857419558001
p_0	1
K	3.09

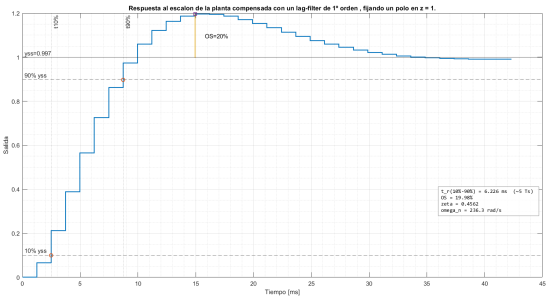


Figura 9: Respuesta al escalón del primer intento de compensador.

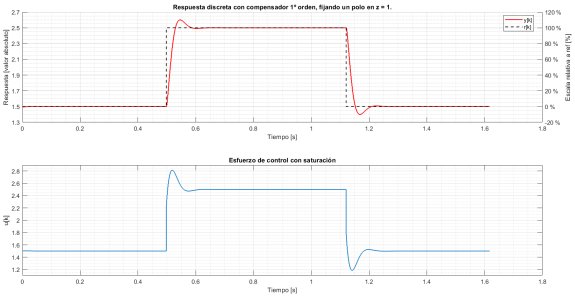


Figura 10: Respuesta al escalón del sistema en lazo cerrado con el segundo compensador diseñado, junto con el esfuerzo de control aplicado por el compensador.

Esto ocurre porque la ubicación de los polos del sistema final no presenta un par de polos dominantes, por lo que la aproximación de segundo orden utilizada al inicio no resulta válida.

Finalmente, optamos por un compensador de segundo orden. En primer lugar, colocamos nuevamente el polo en $z = 1$, pero esta vez anulamos el polo más cercano de la planta (el más lento) y calculamos su contribución a la compensación (valores).

A continuación, diseñamos un segundo compensador para completar lo faltante, anulando el polo más rápido de la planta con un cero en esa ubicación y hallando el nuevo polo mediante el mismo procedimiento aplicado en el primer compensador.

Determinando la ganancia K como en el caso anterior, obtenemos el siguiente resultado, el cual ahora cumple con las condiciones deseadas:

Cuadro III: Parámetros del Compensador 3

$z_{0,0}$	0.9333
$p_{0,0}$	1
$z_{0,1}$	0.436
$p_{0,1}$	0.71135
K	1.14

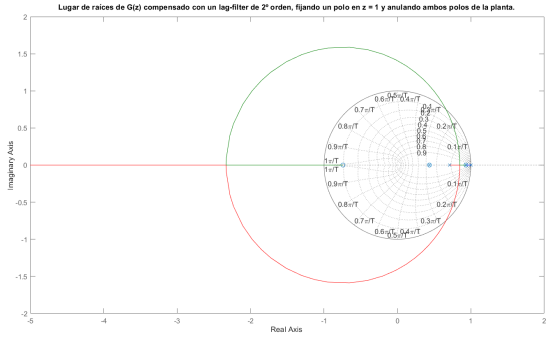


Figura 11: Lugar de raíces de la planta compensada con un filtro *lag* de segundo orden y polo en $z = 1$.

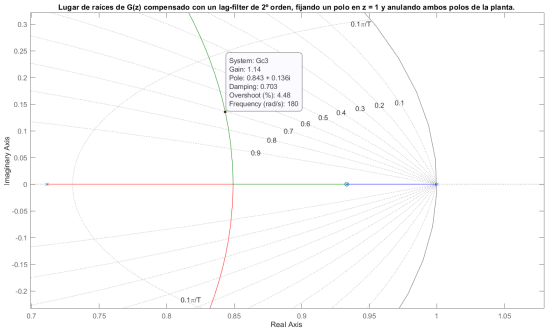


Figura 12: Lugar de raíces de la planta compensada con un filtro *lag* de segundo orden y polo en $z = 1$ (detalle ampliado).

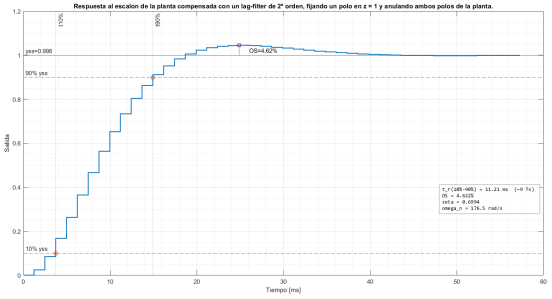


Figura 13: Respuesta al escalón con el tercer compensador propuesto.

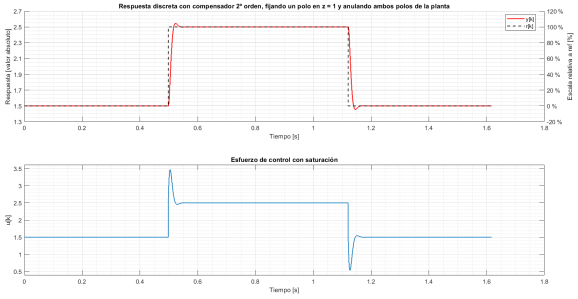
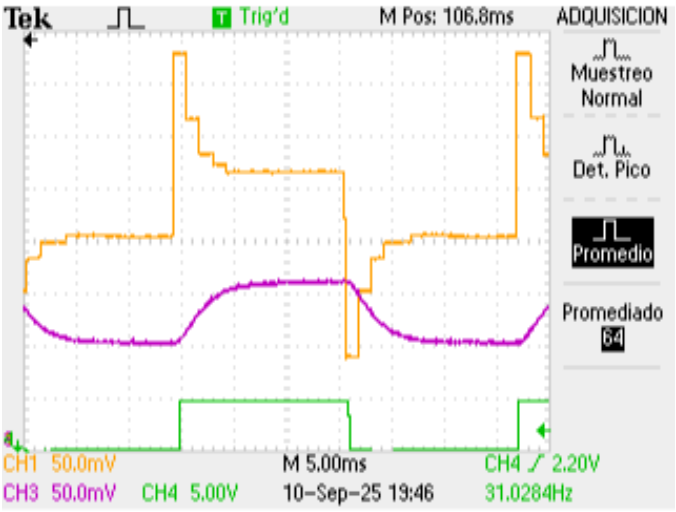


Figura 14: Respuesta al escalón del sistema en lazo cerrado con el tercer compensador diseñado, junto con el esfuerzo de control aplicado por el compensador.

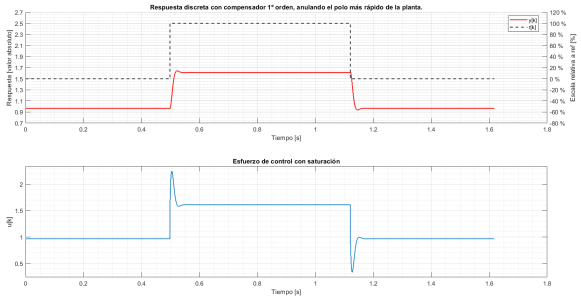
4.1. Implementar las ecuaciones en diferencias:

4.1.4.1. Con el compensador de la sección 3.1.

Se realizó la implementación en *PSoC Creator* empleando los parámetros obtenidos en Matlab para un amortiguamiento objetivo de $\zeta = 0.7$. La Figura 15a muestra el primer compensador y la respuesta medida en placa. El código para el PSoC permite modificar los valores de referencia mínima y máxima, ajustar el período de la señal de referencia y conmutar entre lazo abierto y lazo cerrado, con el fin de realizar las pruebas correspondientes.



(a) Primer compensador diseñado para $\zeta = 0.7$.



(b) Step y esfuerzo del compensador diseñado para $\zeta = 0.7$.

Figura 15

En la Figura 15a se observa un pico pronunciado en el esfuerzo de control (traza amarilla), de manera consistente con lo obtenido en la simulación. La salida (traza magenta) presenta un comportamiento muy similar al simulado (Figura 15b), manteniéndose dentro del rango establecido y logrando un adecuado seguimiento de la referencia.

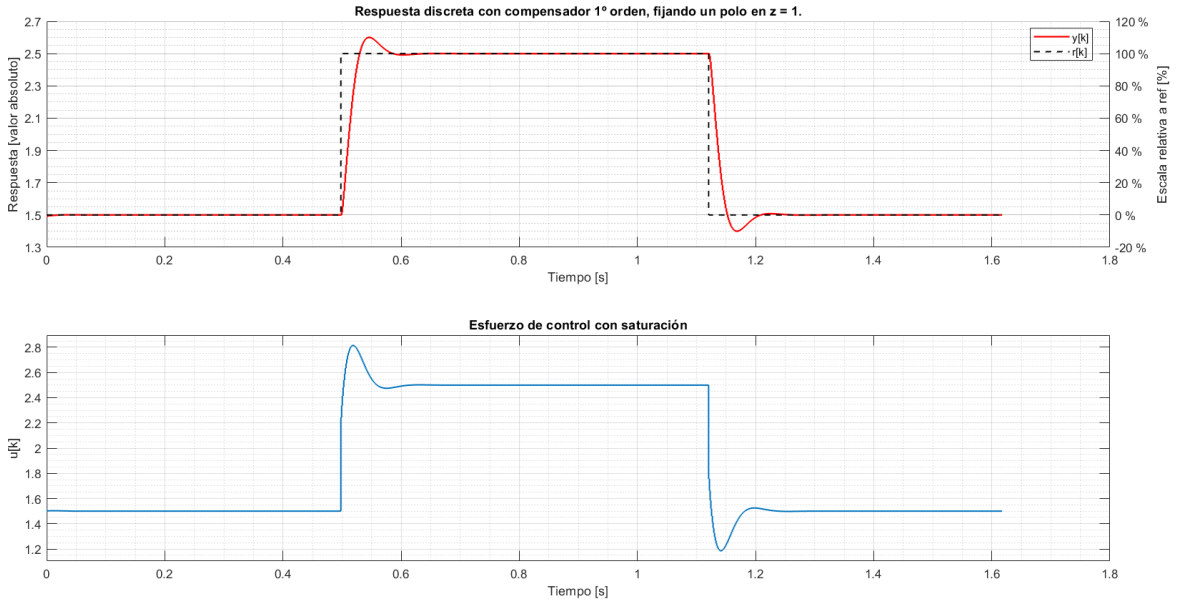
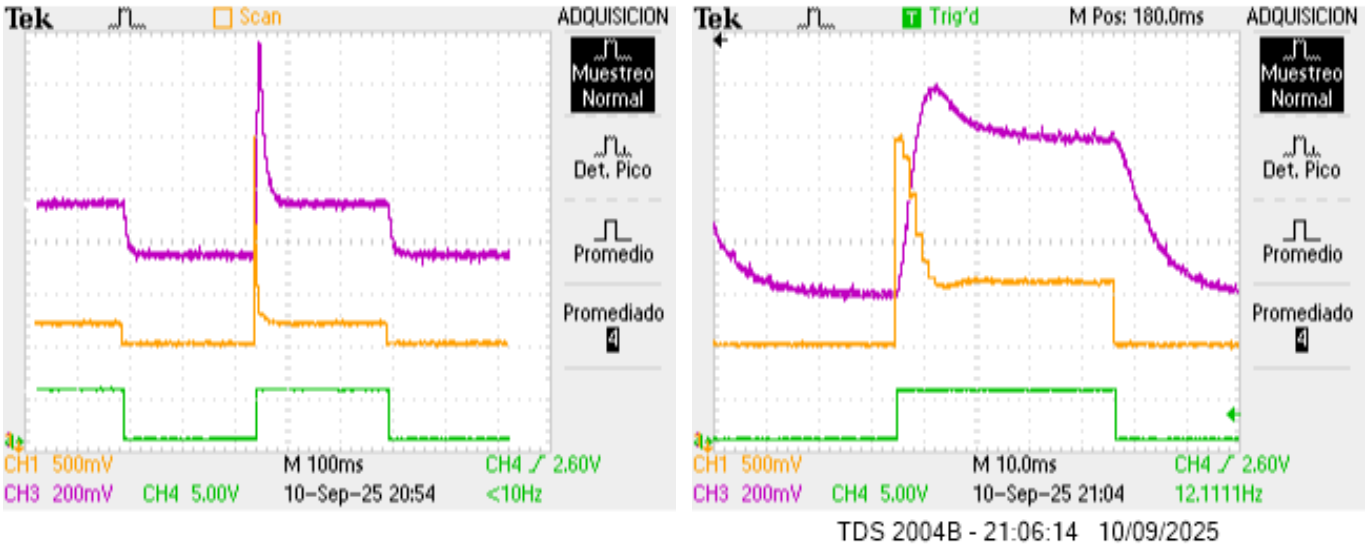


Figura 16: Segundo compensador diseñado para $\zeta = 0.7$.



(a) Primer compensador diseñado para $\zeta = 0.7$ con $ESS = 0$ y un polo en el circulo unitario.

(b) Segundo compensador diseñado para $\zeta = 0.7$ con $ESS = 0$ y un polo cercano al circulo unitario.

Figura 17

4.1.4.2. Con el compensador de la sección 3.2.

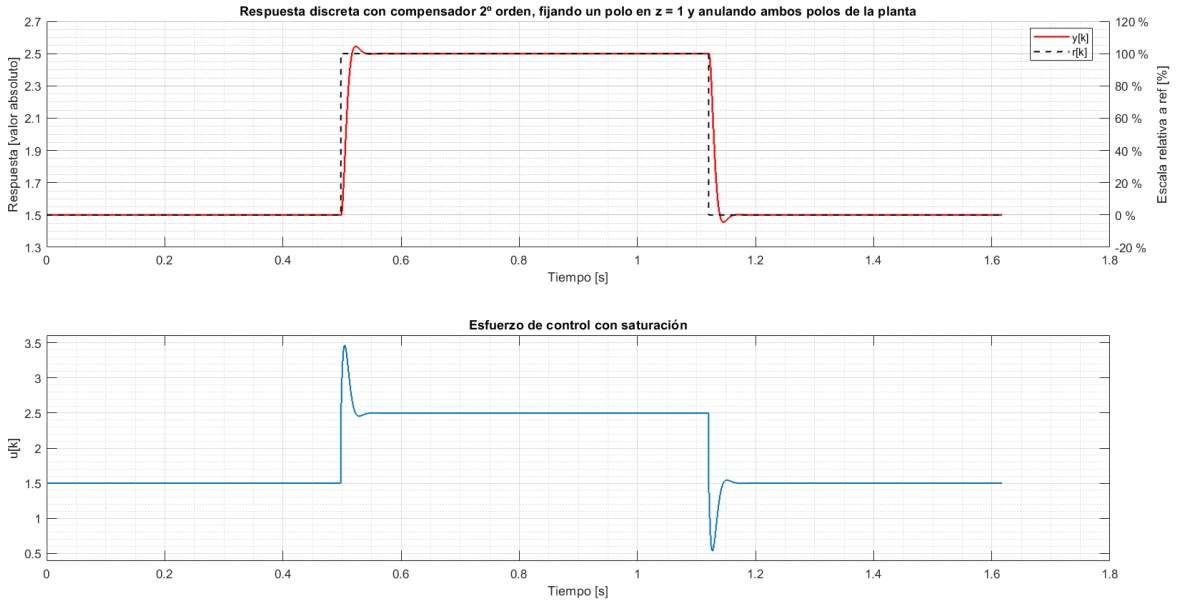
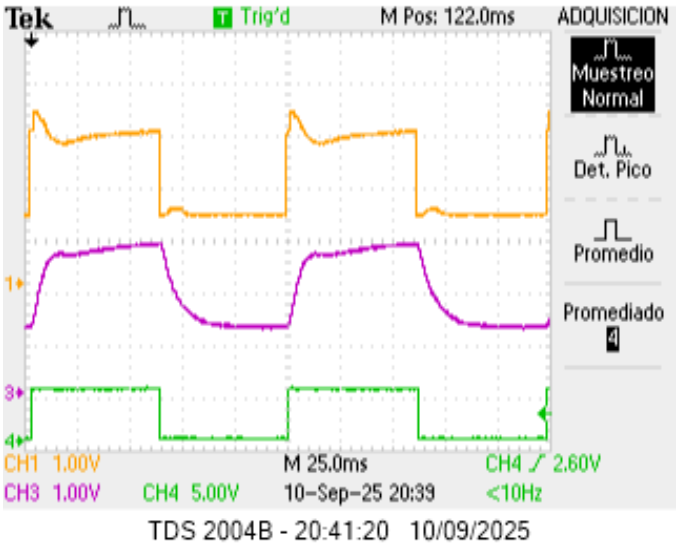
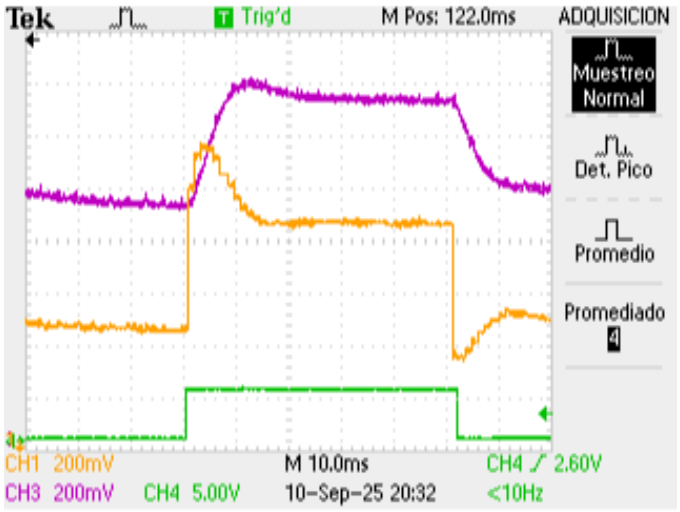


Figura 18: Tercer compensador diseñado en Matlab.



(a) Tercer compensador diseñado para $\zeta = 0.7$ con $ESS = 0$ y un polo cercano en el circulo unitario.



(b) Tercer compensador diseñado para $\zeta = 0.7$ con $ESS = 0$ y un polo cercano en el circulo unitario.

Figura 19

En la Figura 19a se observa que el caso con el polo ubicado cerca del círculo unitario presenta picos en la respuesta, ya que esta ubicación acelera la dinámica del sistema. En contraste, el sistema con el polo en el círculo unitario (Figura 19b) muestra una respuesta más lenta y menos enérgica, producto de un mayor amortiguamiento. Por otro lado, las imágenes de implementación presentadas en la Figura 19 no reproducen con fidelidad el comportamiento de la simulación mostrada en la Figura 18. Esta discrepancia se atribuye a un modelado inadecuado de la planta o del compensador, lo que introduce diferencias notorias entre la simulación teórica y la respuesta medida en hardware.

VI-E. Nuevo Modelado de la Planta

Con el objetivo de mejorar las prestaciones y evitar las limitaciones derivadas de las tolerancias de los componentes pasivos (resistencias y capacitores de baja precisión), se decidió realizar un modelado alternativo de la planta utilizando MATLAB.

Para ello, se empleó el osciloscopio a fin de muestrear la respuesta del sistema ante una entrada escalón. Posteriormente, mediante las funciones de identificación de sistemas en MATLAB, se llevó a cabo una estimación de segundo orden que aproxima de manera adecuada el comportamiento dinámico de la planta.

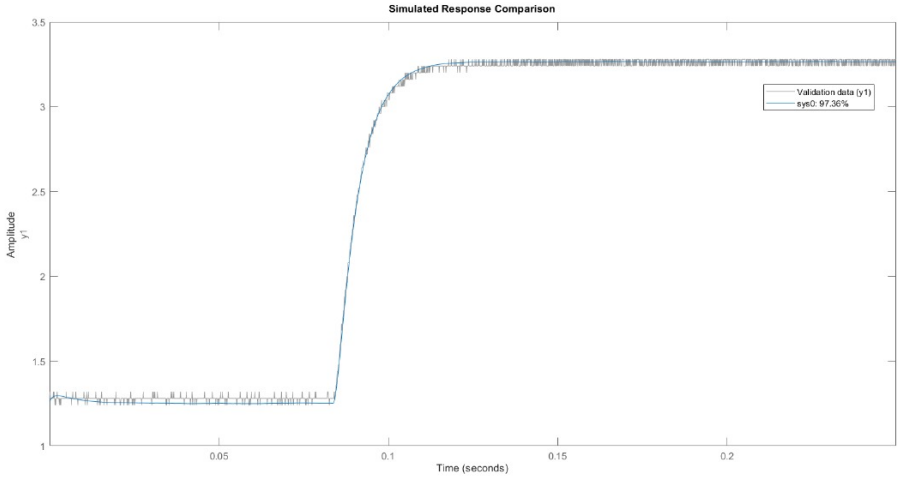


Figura 20: Respuesta al escalón de la planta estimada mediante identificación en MATLAB.

De esta estimación se obtuvo la siguiente función de transferencia característica del sistema identificado:

$$G(s) \;=\; \frac{1.2472e05}{(s+771.3)(s+157.3)}, \tag{15}$$

donde los parámetros K, a y b fueron ajustados automáticamente por el algoritmo de estimación en función de los datos experimentales.

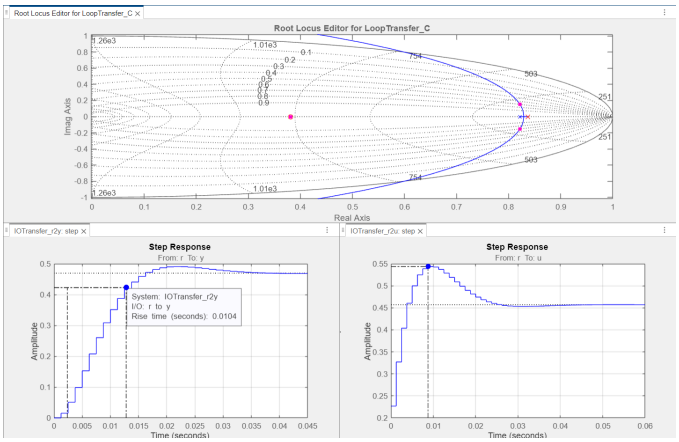
A partir de este nuevo modelo, y siguiendo procedimientos análogos a los presentados en secciones anteriores, se diseñaron dos compensadores que fueron implementados de manera similar al caso inicial.

Cuadro IV: Parámetros del Compensador 1 (con la nueva planta)

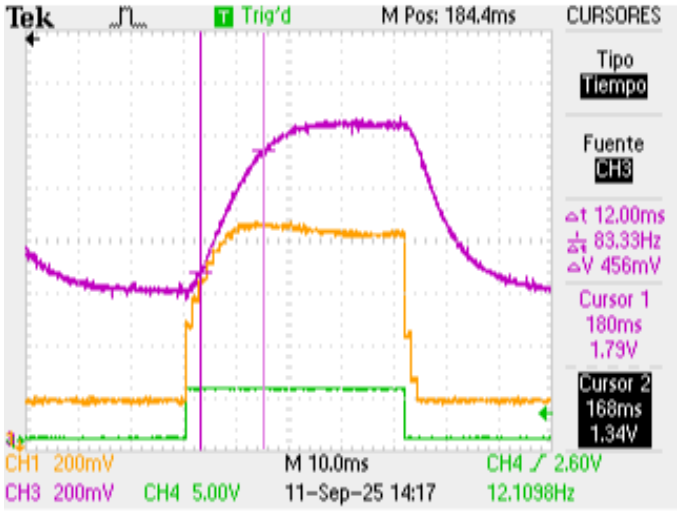
z_0	0.380927627742363
p_0	0.836970379456654
K	0.226961597782889

Cuadro V: Parámetros del Compensador 2 (con la nueva planta)

$z_{0,0}$	0.380927627742363
$z_{0,1}$	0.821584532561100
$p_{0,0}$	0.753973536568200
$p_{0,1}$	1
K	0.287529112579611

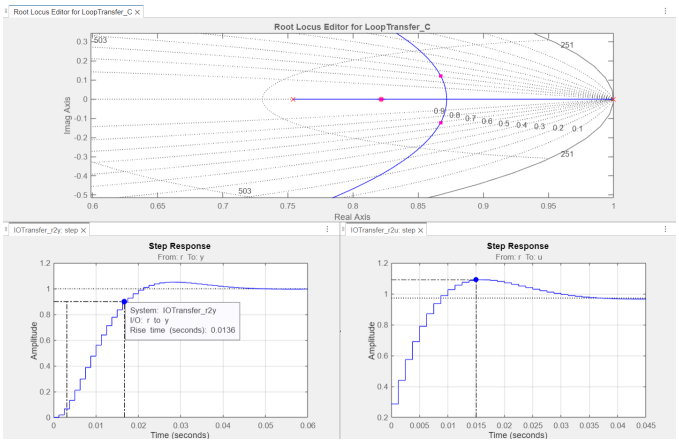


(a) Version dos del primer compensador diseñado en Matlab para $\zeta = 0.7$.

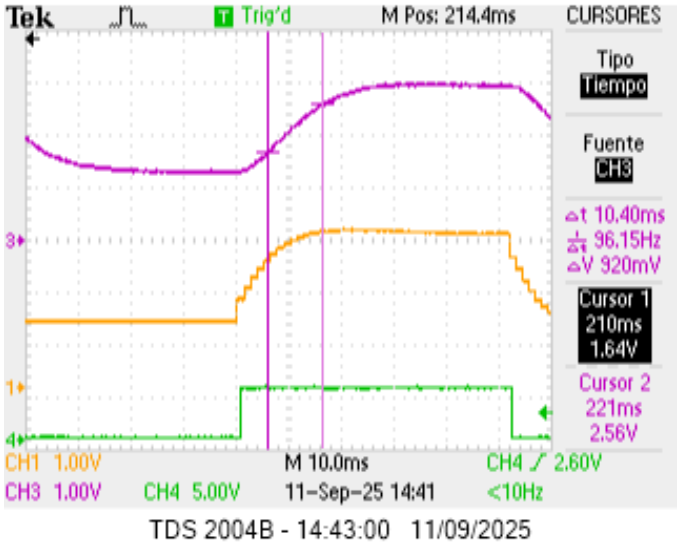


(b) Implementación del compensador en el PSoC.

Figura 21



(a) Version dos del segundo compensador diseñado en Matlab para $\zeta = 0.7$ y $ESS = 0$.



(b) Implementación del compensador en el PSoC.

Figura 22

VI-F. Resultados

Cuadro VI: Comparación entre cálculos y mediciones

Compensador	**Overshoot (%)**	t_r **(ms)**	ζ
Compensador 1 (primer modelo) - Cálculo	4.79	11.21	0.69
Compensador 1 (primer modelo) - Medición	NV	5	> 0.6
Compensador 3 (primer modelo) - Cálculo	4.62	11.21	0.69
Compensador 3 (primer modelo) - Medición (polo en 1)	300	NV	NE
Compensador 3 (primer modelo) - Medición (polo en 0.9)	25	10	0.404
Compensador 1 (segundo modelo) - Cálculo	5	10.4	0.70
Compensador 1 (segundo modelo) - Medición	5	7	0.69
Compensador 2 (segundo modelo) - Cálculo	5	13.6	0.70
Compensador 2 (segundo modelo) - Medición	5	8	0.69

VI-G. Conclusión

El diseño de los compensadores, si bien logró estabilizar el sistema, no reprodujo exactamente la respuesta esperada según los cálculos iniciales. En particular, los valores de ζ y del tiempo de subida no coincidieron con lo estimado teóricamente. Esto se debió principalmente a un modelado poco preciso de la planta, sumado al uso de valores nominales de resistencias y capacitores, lo cual introdujo un error significativo en la implementación práctica.

Además, se observó que al ubicar un polo cercano a $z = 1$ la respuesta tiende a desestabilizarse; sin embargo, con un ajuste fino de su posición se logró mejorar el desempeño. Este efecto puede atribuirse también a la incertidumbre en el muestreo y a las pequeñas variaciones en el *clock*, que desplazan el polo fuera del círculo unitario.

Finalmente, al emplear un modelo más preciso de la planta, sin necesidad de modificar de manera considerable los valores de K, polos ni ceros, se obtuvieron respuestas muy cercanas a las calculadas en Matlab. Esto demuestra la importancia de un modelado adecuado como base para una correcta implementación del control en hardware.

```
close all
clear all

addpath('..\Lab1\')

%% Definicion de parametros
R_1 = 15e3;
R_3 = 15e3;
C_2 = 100e-9;

R_2 = 82e3;
R_4 = 82e3;
C_1 = 0.22e-6;

%% Generar funcion de transferencia d
numStage = [-R_3/R_1 -R_4/R_2];
denStage = { [C_2*R_3 1], [C_1*R_4 1] };

% Usamos celdas para guardar los tf de cada stage
Gstage = cell(1,2);
G = 1;
for i = 1:2
Gstage{i} = tf(numStage(i), denStage{i});
G = G*Gstage{i};
end

%% Analizamos en el tiempo
[tr, ts, wn] = plot_step_info(G);
disp('Planta continua G(s):')
G
zpk(G)



%% Paso 1: Definir periodo de muestreo
% Se busca una mejora de 4 en el tiempo de rising , o sea tr = 10 ms
N = 4;
T = tr/(8*N);
%Gcl = feedback(G,1);
%figure;
%step(Gcl)
%T = stepinfo(Gcl).SettlingTime/8;

%% Paso 2: Digitalizar con ZOH
Gd = c2d(G, T, 'zoh');
disp('Planta digital G(z):')
Gd
zpk(Gd)
figure;
pzmap(Gd);
title('Lugar de raíces de G(z).')

zgrid;
z = tf([1 0],1,T);

%% Paso 3: Análisis en tiempo discreto lazo abierto
figure;

% Obtenemos salida y tiempo de la función step
[y, t] = step(Gd);

% Graficamos con stairs (propio de señales discretas)
stairs(t*1000, y, 'LineWidth',1.4);

```

```
title('Respuesta al escalón de la planta digitalizad')
xlabel('Tiempo [ms]');
ylabel('Salida');
grid on;
ax = gca;
ax.XMinorTick = 'on';   % activamos minor ticks
ax.YMinorTick = 'on';
grid minor;

%% Paso 4: Definimimos los valores de la simulación
umin = 0;
umax =  4.08;
refmin = 1.5;
refmax = 2.5;
n_per_seg = 500;
%% Paso 5: Lugar de raíces para zita = 0.7, compensador P
%figure

%rlocus(Gd)
%zgrid     % agrega la grilla en el plano-z
z0=0.436;
p0=0.769221;%0.7705;
C_1 = (z-z0)/(z-p0);
figure;
Gc1 = Gd*C_1;
rlocus(Gc1);
title('Lugar de raíces de G(z) compensado con un lag-filter de 1° orden, anulando el polo más rápido de la planta.')

zgrid     % agrega la grilla en el plano-z
%pause;
%[K, ~] = rlocfind(Gc1);    % hacés click donde querés los polos
%K = 0.73139;
%K = 0.82543;
K = 0.743;
Gc1f = feedback(K*Gc1,1);
%figure;
%step(Gc1f)
info = stepinfo(Gc1f);
info

info_ext = plot_step_annot(Gc1f, 'la planta compensada con un lag-filter de 1° orden, anulando el polo más rápido de la planta.');
%step(Gc3f);
info = stepinfo(Gc1f);
info
info_ext

[td, refd, yd, ud, ed, coefs] = sim_compensador_first_order( ...
Gd, z0, p0, K, T, umin, umax, refmin, refmax, n_per_seg);

Nini = 100;               % descartar al inicio
Nfin = 100;               % descartar al final
idx0 = Nini + 1;          % índice inicial válido
idx1 = length(td) - Nfin;  % índice final válido

td   = td(idx0:idx1)-td(idx0);
refd = refd(idx0:idx1);
yd   = yd(idx0:idx1);
ud   = ud(idx0:idx1);
ed   = ed(idx0:idx1);

% Si querés que el tiempo arranque en 0
```

```
130
131     figure;
132
133     % ---------- Subplot 1 ----------
134     axAbs = subplot(2,1,1);   % eje izquierdo (absoluto)
135     hold(axAbs,'on'); grid(axAbs,'on');
136
137     % y[k] en rojo (abs)
138     hY = stairs(axAbs, td, yd, 'r-', 'LineWidth', 1.4);
139
140     % Armamos eje derecho transparente
141     axPct = axes('Position', get(axAbs,'Position'), ...
142     'Color','none', 'YAxisLocation','right', ...
143     'XLim', get(axAbs,'XLim'), 'XTick',[], 'Box','off');
144     hold(axPct,'on');
145
146     % r[k] en negro, graficado en %
147     ref_pct = 100*(refd - refmin)/(refmax - refmin);
148     hR = stairs(axPct, td, refd, 'k--', 'LineWidth', 1.2);
149
150     % ======= Cálculo de límites con margen =======
151     yd_pct  = 100*(yd - refmin)/(refmax - refmin);
152     pctAll  = [yd_pct; ref_pct];
153
154     % Valores extremos en % con margen del 5 %
155     rawMin = min(pctAll);
156     rawMax = max(pctAll);
157     span   = rawMax - rawMin;
158     pctMin = rawMin - 0.05*span;
159     pctMax = rawMax + 0.05*span;
160
161     % Redondeamos a múltiplos de 20 para ticks
162     stepPct = 20;
163     tickMin = stepPct*floor(pctMin/stepPct);
164     tickMax = stepPct*ceil (pctMax/stepPct);
165     pctTicks = tickMin:stepPct:tickMax;
166
167     % Convertimos a absolutos
168     valTicks = refmin + (pctTicks/100)*(refmax - refmin);
169
170     % Aplicamos a ambos ejes
171     set(axAbs,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks);
172     set(axPct,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks,...
173     'YTickLabel',compose('%.0f %%',pctTicks));
174
175     % Etiquetas
176     xlabel(axAbs,'Tiempo [s]');
177     ylabel(axAbs,'Respuesta [valor absoluto]');
178     ylabel(axPct,'Escala relativa a ref [%]');
179     title(axAbs,'Respuesta discreta con compensador 1° orden, anulando el polo más rápido de
    la planta.');
180     legend(axAbs, [hY, hR], {'y[k]', 'r[k]'}, 'Location', 'best');
181
182     % ---------- Subplot 2: esfuerzo de control ----------
183     axU = subplot(2,1,2);
184     stairs(axU, td, ud, 'LineWidth',1.2); grid(axU,'on');
185     yline(axU, umax,'r:'); yline(axU, umin,'r:');
186     xlabel(axU,'Tiempo [s]'); ylabel(axU,'u[k]');
187     title(axU,'Esfuerzo de control con saturación');
188
189     % --- Margen de 5% en eje Y ---
190     uMin = min(ud);
191     uMax = max(ud);
192     span = uMax - uMin;
193     ylim(axU, [uMin - 0.05*span, uMax + 0.05*span]);
194
195
```

```
196     grid(axAbs,'on');     % grilla principal
197     grid(axAbs,'minor');  % grilla secundaria
198
199     grid(axPct,'on');
200     grid(axPct,'minor');
201
202     grid(axU,'on');
203     grid(axU,'minor');
204
205     %% Paso 6: Lugar de raices para zita = 0.7 y y ESS = 0, compensador PI
206     z0 = 0.857419558001;
207     p0 = 1;
208     C_2 = (z-z0)/(z-p0); %z=0.8433 %8433
209     Gc2 =Gd*C_2;
210     figure;
211     rlocus(Gc2);
212     title('Lugar de raíces de G(z) compensado con un lag-filter de 1° orden , fijando un polo
        en z = 1.')
213
214     zgrid
215     %[K, ~] = rlocfind(Gc2);    % hacés click donde querés los polos
216     %K = 3.2779;
217     K2 = 3.09;
218     %figure;
219     Gc2f= feedback(K2*Gc2,1);
220     %step(Gc2f);
221     info = stepinfo(Gc2f);
222     info
223
224     info_ext = plot_step_annot(Gc2f, 'la planta compensada con un lag-filter de 1° orden ,
        fijando un polo en z = 1.');
225     %step(Gc3f);
226     info = stepinfo(Gc2f);
227     info
228     info_ext
229
230
231     [td, refd, yd, ud, ed, coefs] = sim_compensador_first_order( ...
232     Gd, z0, p0, K, T, umin, umax, refmin, refmax, n_per_seg);
233
234
235     Nini = 100;               % descartar al inicio
236     Nfin = 100;               % descartar al final
237     idx0 = Nini + 1;          % índice inicial válido
238     idx1 = length(td) - Nfin; % índice final válido
239
240     td   = td(idx0:idx1)-td(idx0);
241     refd = refd(idx0:idx1);
242     yd   = yd(idx0:idx1);
243     ud   = ud(idx0:idx1);
244     ed   = ed(idx0:idx1);
245
246
247
248
249     figure;
250
251     % ---------- Subplot 1 ----------
252     axAbs = subplot(2,1,1);   % eje izquierdo (absoluto)
253     hold(axAbs,'on'); grid(axAbs,'on');
254
255     % y[k] en rojo (abs)
256     hY = stairs(axAbs, td, yd, 'r-', 'LineWidth', 1.4);
257
258     % Armamos eje derecho transparente
259     axPct = axes('Position', get(axAbs,'Position'), ...
260     'Color','none', 'YAxisLocation','right', ...
```

```
    'XLim', get(axAbs,'XLim'), 'XTick',[], 'Box','off');
    hold(axPct,'on');

    % r[k] en negro, graficado en %
    ref_pct = 100*(refd - refmin)/(refmax - refmin);
    hR = stairs(axPct, td, refd, 'k--', 'LineWidth', 1.2);

    % ======= Cálculo de límites con margen =======
    yd_pct  = 100*(yd - refmin)/(refmax - refmin);
    pctAll  = [yd_pct; ref_pct];

    % Valores extremos en % con margen del 5 %
    rawMin = min(pctAll);
    rawMax = max(pctAll);
    span   = rawMax - rawMin;
    pctMin = rawMin - 0.05*span;
    pctMax = rawMax + 0.05*span;

    % Redondeamos a múltiplos de 20 para ticks
    stepPct = 20;
    tickMin = stepPct*floor(pctMin/stepPct);
    tickMax = stepPct*ceil (pctMax/stepPct);
    pctTicks = tickMin:stepPct:tickMax;

    % Convertimos a absolutos
    valTicks = refmin + (pctTicks/100)*(refmax - refmin);

    % Aplicamos a ambos ejes
    set(axAbs,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks);
    set(axPct,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks,...
    'YTickLabel',compose('%.0f %%',pctTicks));

    % Etiquetas
    xlabel(axAbs,'Tiempo [s]');
    ylabel(axAbs,'Respuesta [valor absoluto]');
    ylabel(axPct,'Escala relativa a ref [%]');
    title(axAbs,'Respuesta discreta con compensador 1° orden, fijando un polo en z = 1.');
    legend(axAbs, [hY, hR], {'y[k]', 'r[k]'}, 'Location', 'best');

    % ---------- Subplot 2: esfuerzo de control ----------
    axU = subplot(2,1,2);
    stairs(axU, td, ud, 'LineWidth',1.2); grid(axU,'on');
    yline(axU, umax,'r:'); yline(axU, umin,'r:');
    xlabel(axU,'Tiempo [s]'); ylabel(axU,'u[k]');
    title(axU,'Esfuerzo de control con saturación');

    % --- Margen de 5% en eje Y ---
    uMin = min(ud);
    uMax = max(ud);
    span = uMax - uMin;
    ylim(axU, [uMin - 0.05*span, uMax + 0.05*span]);

    grid(axAbs,'on');     % grilla principal
    grid(axAbs,'minor');  % grilla secundaria

    grid(axPct,'on');
    grid(axPct,'minor');

    grid(axU,'on');
    grid(axU,'minor');



    %% Paso 7: Lugar de raices para zita = 0.7 y y ESS = 0, compensador PI de segundo orden
    z0_0 = 0.9333;
    p0_0 = 1;
    z0_1 = 0.436;
```

```
p0_1 = 0.71135;
C_3 = ((z-z0_0)*(z-z0_1))/((z-p0_0)*(z-p0_1));
Gc3 =Gd*C_3;  %z=0.8433  %8433
figure;
rlocus(Gc3)
title('Lugar de raíces de G(z) compensado con un lag-filter de 2° orden, fijando un polo en z = 1 y anulando ambos polos de la planta.');

zgrid
%pause;
%[K, ~] = rlocfind(Gc3);    % hacés click donde querés los polos
K = 1.14;
%figure;
Gc3f= feedback(K*Gc3,1);

info_ext = plot_step_annot(Gc3f, 'la planta compensada con un lag-filter de 2° orden, fijando un polo en z = 1 y anulando ambos polos de la planta.');
%step(Gc3f);
info = stepinfo(Gc3f);
info
info_ext


% --- Llamada a tu simulador (misma interfaz que tu first_order) ---
[td, refd, yd, ud, ed, coefs] = sim_compensador_second_order( ...
Gd, z0_0, z0_1, p0_0, p0_1, K, T, umin, umax, refmin, refmax, n_per_seg);


Nini = 100;               % descartar al inicio
Nfin = 100;               % descartar al final
idx0 = Nini + 1;          % índice inicial válido
idx1 = length(td) - Nfin;  % índice final válido

td   = td(idx0:idx1)-td(idx0);
refd = refd(idx0:idx1);
yd   = yd(idx0:idx1);
ud   = ud(idx0:idx1);
ed   = ed(idx0:idx1);




figure;

% ---------- Subplot 1 ----------
axAbs = subplot(2,1,1);   % eje izquierdo (absoluto)
hold(axAbs,'on'); grid(axAbs,'on');

% y[k] en rojo (abs)
hY = stairs(axAbs, td, yd, 'r-', 'LineWidth', 1.4);

% Armamos eje derecho transparente
axPct = axes('Position', get(axAbs,'Position'), ...
'Color','none', 'YAxisLocation','right', ...
'XLim', get(axAbs,'XLim'), 'XTick',[], 'Box','off');
hold(axPct,'on');

% r[k] en negro, graficado en %
ref_pct = 100*(refd - refmin)/(refmax - refmin);
hR = stairs(axPct, td, refd, 'k--', 'LineWidth', 1.2);

% ======= Cálculo de límites con margen =======
yd_pct  = 100*(yd - refmin)/(refmax - refmin);
pctAll  = [yd_pct; ref_pct];

% Valores extremos en % con margen del 5 %
rawMin = min(pctAll);
```

```
    rawMax = max(pctAll);
    span   = rawMax - rawMin;
    pctMin = rawMin - 0.05*span;
    pctMax = rawMax + 0.05*span;

    % Redondeamos a múltiplos de 20 para ticks
    stepPct = 20;
    tickMin = stepPct*floor(pctMin/stepPct);
    tickMax = stepPct*ceil (pctMax/stepPct);
    pctTicks = tickMin:stepPct:tickMax;

    % Convertimos a absolutos
    valTicks = refmin + (pctTicks/100)*(refmax - refmin);

    % Aplicamos a ambos ejes
    set(axAbs,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks);
    set(axPct,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks,...
    'YTickLabel',compose('%.0f %%',pctTicks));

    % Etiquetas
    xlabel(axAbs,'Tiempo [s]');
    ylabel(axAbs,'Respuesta [valor absoluto]');
    ylabel(axPct,'Escala relativa a ref [%]');
    title(axAbs,'Respuesta discreta con compensador 2° orden, fijando un polo en z = 1 y
    anulando ambos polos de la planta');
    legend(axAbs, [hY, hR], {'y[k]', 'r[k]'}, 'Location', 'best');

    % ---------- Subplot 2: esfuerzo de control ----------
    axU = subplot(2,1,2);
    stairs(axU, td, ud, 'LineWidth',1.2); grid(axU,'on');
    yline(axU, umax,'r:'); yline(axU, umin,'r:');
    xlabel(axU,'Tiempo [s]'); ylabel(axU,'u[k]');
    title(axU,'Esfuerzo de control con saturación');

    % --- Margen de 5% en eje Y ---
    uMin = min(ud);
    uMax = max(ud);
    span = uMax - uMin;
    ylim(axU, [uMin - 0.05*span, uMax + 0.05*span]);









    grid(axAbs,'on');     % grilla principal
    grid(axAbs,'minor');  % grilla secundaria

    grid(axPct,'on');
    grid(axPct,'minor');

    grid(axU,'on');
    grid(axU,'minor');


    %% impresion antigua
    % % --- Gráficas estilo informe ---
    % figure;
    % subplot(2,1,1);
    % stairs(td, refd, 'k--','LineWidth',1.0); hold on;
    % stairs(td, yd,   'LineWidth',1.4);
    % grid on; xlabel('Tiempo [s]'); ylabel('Respuesta de la planta/Referencia');
    % title('Respuesta discreta con compensador 2° orden'); legend('r[k]','y[k]','Location','
    best');
    %
    % subplot(2,1,2);
```

```
% stairs(td, ud, 'LineWidth',1.2);
% yline(umax,'r:'); yline(umin,'r:');
% grid on; xlabel('Tiempo [s]'); ylabel('u[k]');
% title('Esfuerzo de control con saturación');
```

Listing 1: Primera hoja de cálculos utilizada.

```
function info_ext = plot_step_annot(sys, nameStr, Ts_override)
% STEP discreto con STAIRS; anota 10-90% (en s y en T), OS, zeta y omega_n.
% Grafica en milisegundos y marca el OS en el pico.

if nargin < 2 || isempty(nameStr), nameStr = inputname(1); end
if isempty(nameStr), nameStr = 'sys'; end

% ----- Respuesta y métricas básicas -----
[y, t] = step(sys); y = y(:); t = t(:);
t_ms = t * 1000;    % graficamos en milisegundos

% Estimar yss por promedio en cola
Ntail = max(10, round(0.05*length(y)));
yss = mean(y(end-Ntail+1:end));

% Límites 10-90
if abs(yss) < 1e-12
t10 = NaN; t90 = NaN; tr_10_90 = NaN; y10 = 0; y90 = 0;
else
y10 = 0.10*yss; y90 = 0.90*yss;
sgn = sign(yss);
idx10 = find(sgn*y >= sgn*y10, 1, 'first');
idx90 = find(sgn*y >= sgn*y90, 1, 'first');
t10 = tern(~isempty(idx10), t(idx10), NaN);
t90 = tern(~isempty(idx90), t(idx90), NaN);
tr_10_90 = tern(~isnan(t10)&&~isnan(t90)&&t90>=t10, t90 - t10, NaN);
end
t10_ms = t10 * 1000; t90_ms = t90 * 1000;

% Pico y %OS (para zeta y omega_n)
[ypeak_raw, idxpk] = max(sign(yss).*y);
ypeak = sign(yss)*ypeak_raw;
tpeak = t(idxpk); tpeak_ms = tpeak * 1000;
OS_percent = tern(yss ~= 0, max(0, (ypeak - yss)/abs(yss)*100), NaN);

% zeta desde %OS:  OS% = 100*exp(-zeta*pi/sqrt(1-zeta^2))
if isnan(OS_percent) || OS_percent <= 0
zeta_est = NaN;
else
logterm = log(OS_percent/100);
zeta_est = -logterm / sqrt(pi^2 + logterm^2);
end

% ----- Ts y rise en T -----
Ts = NaN;
try
if isprop(sys,'Ts') && ~isempty(sys.Ts) && sys.Ts > 0, Ts = sys.Ts; end
catch, end
if nargin >= 3 && ~isempty(Ts_override) && Ts_override > 0, Ts = Ts_override; end
tr_10_90_T = tern(~isnan(Ts) && ~isnan(tr_10_90), tr_10_90/Ts, NaN);

% ----- Estimacion de omega_n -----
omega_n = NaN;
if ~isnan(zeta_est) && zeta_est < 1 && OS_percent > 0 && tpeak > 0
omega_n = pi / ( tpeak * sqrt(1 - zeta_est^2) );
end
if isnan(omega_n)
S = stepinfo(y, t);    % Ts(2%) ~ 4/(zeta*omega_n)
if isfield(S,'SettlingTime') && ~isempty(S.SettlingTime) && S.SettlingTime > 0 && ~isnan(
zeta_est) && zeta_est > 0
```

```
    omega_n = 4 / ( zeta_est * S.SettlingTime );
    end
    end
    if isnan(omega_n) && ~isnan(tr_10_90) && tr_10_90 > 0
    k_rise = 1.4;   % aprox para 10-90% en 2do orden subamortiguado
    omega_n = k_rise / tr_10_90;
    end

    % ----- Grafico -----
    figure;
    stairs(t_ms, y, 'LineWidth', 1.5); grid on;grid minor; hold on;
    xlabel('Tiempo [ms]'); ylabel('Salida');
    title(sprintf('Respuesta al escalon de %s', nameStr));

    % Líneas guía 10% y 90% y marcadores (en ms)
    if ~isnan(tr_10_90)
    yline(y10, '--', '10% yss', 'LabelHorizontalAlignment','left');
    yline(y90, '--', '90% yss', 'LabelHorizontalAlignment','left');
    xline(t10_ms, ':', 't10%');
    xline(t90_ms, ':', 't90%');
    plot([t10_ms t90_ms], [y10 y90], 'o', 'MarkerSize', 6, 'LineWidth', 1.2);
    end
    % Línea de yss
    yline(yss, '-', sprintf('yss=%.3g', yss), 'LabelHorizontalAlignment','left');

    % ----- Marca del OS en el pico -----
    if ~isnan(OS_percent) && OS_percent > 0
    % línea vertical desde yss a ypeak en tpeak
    plot([tpeak_ms tpeak_ms], [yss ypeak], '-', 'LineWidth', 1.2);
    % texto al costado derecho de la línea
    dx = 0.02 * (t_ms(end) - t_ms(1));
    text(tpeak_ms + dx, yss + 0.5*(ypeak - yss), ...
    sprintf('OS=%.3g%%', OS_percent), ...
    'Interpreter','none', 'Margin',2);
    % marcar el pico
    plot(tpeak_ms, ypeak, 's', 'MarkerSize', 6, 'LineWidth', 1.2);
    end

    % ----- Cuadro de metricas (abajo-derecha; chico) -----
    w = 0.15; h = 0.09; rmargin = 0.1; x = 1 - rmargin - w; y0 = 0.30;
    txtLines = {
        sprintf('t_r(10%%-90%%) = %.4g ms  (~%.4g Ts)', safeNum(tr_10_90*1000), safeNum(
    tr_10_90_T))
        sprintf('OS = %.4g%%',          safeNum(OS_percent))
        sprintf('zeta = %.4g',          safeNum(zeta_est))
        sprintf('omega_n = %.4g rad/s', safeNum(omega_n))
    };
    annotation('textbox', [x y0 w h], ...
    'String', txtLines, 'Interpreter','none', ...
    'FontName','Consolas', 'FontSize',9, ...
    'EdgeColor',[0.5 0.5 0.5], 'FaceAlpha',0.88, 'BackgroundColor','w');

    hold off;

    % ----- Salida -----
    info_ext = struct( ...
    'yss', yss, ...
    't10', t10, ...
    't90', t90, ...
    'tr_10_90', tr_10_90, ...
    'tr_10_90_T', tr_10_90_T, ...
    'OS_percent', OS_percent, ...
    'zeta_est', zeta_est, ...
    'omega_n', omega_n, ...
    'Ts', Ts);
    end
```

```
    % --------- helpers ---------
    function y = tern(cond, a, b)
    if cond, y = a; else, y = b; end
    end

    function v = safeNum(x)
    if isnan(x) || isinf(x), v = NaN; else, v = x; end
    end
    function [td, refd, yd, ud, ed, coefs] = sim_compensador_first_order( ...
    Gd, z0, p0, Kc, T, umin, umax, refmin, refmax, n_per_seg)

    % ===== Igual que tu PID: y(k+1)=b0*u(k)+b1*u(k-1)-a1*y(k)-a2*y(k-1) =====
    % Controlador (posición): u(k) = p0*u(k-1) + Kc*ed(k) - Kc*z0*ed(k-1)
    % ed(k) = refd(k) - yd(k)

    if nargin < 10 || isempty(n_per_seg), n_per_seg = 600; end
    if ~isa(Gd,'tf'), error('Gd debe ser tf discreto.'); end

    % --- Coefs planta exactamente como en tu ejemplo ---
    [numD, denD] = tfdata(Gd, 'v');
    b0 = numD(2);
    b1 = numD(3);
    a1 = denD(2);
    a2 = denD(3);
    coefs = struct('b0',b0,'b1',b1,'a1',a1,'a2',a2);

    % --- Tiempo y referencia (refmin -> refmax -> refmin) ---
    N  = 3*n_per_seg;
    td = (0:N-1)' * T;
    refd = [refmin*ones(n_per_seg,1);
    refmax*ones(n_per_seg,1);
    refmin*ones(n_per_seg,1)];

    % --- Inicialización idéntica a tu estilo ---
    yd = zeros(N,1);
    ed = zeros(N,1);
    ud = zeros(N,1);

    % --- Loop (misma estructura y orden que tu PID) ---
    for k = 3:N-1
    ed(k) = refd(k) - yd(k);

    % Compensador: u(k) = p0*u(k-1) + Kc*ed(k) - Kc*z0*ed(k-1)
    u_k = p0*ud(k-1) + Kc*ed(k) - Kc*z0*ed(k-1);


    if u_k>umax
    ud(k) = umax;
    elseif u_k<umin
    ud(k) = umin;
    else
    ud(k) = u_k;
    end


    % Planta discreta (dividido por a0 implícito como en tu ejemplo)
    yd(k+1) = b0*ud(k) + b1*ud(k-1) - a1*yd(k) - a2*yd(k-1);
    end
    end

    function [td, refd, yd, ud, ed, coefs] = sim_compensador_second_order( ...
    Gd, z0_0, z0_1, p0_0, p0_1, Kc, T, umin, umax, refmin, refmax, n_per_seg)

    % ===== Planta: y(k+1)=b0*u(k)+b1*u(k-1)+b2*u(k-2)-a1*y(k)-a2*y(k-1) =====
    % Compensador (posición, 2° orden):
    % u(k) = (p0_0+p0_1)*u(k-1) - (p0_0*p0_1)*u(k-2) ...
    %      + Kc*[ e(k) - (z0_0+z0_1)*e(k-1) + (z0_0*z0_1)*e(k-2) ]
```

```
    % e(k) = refd(k) - yd(k)

    if nargin < 12 || isempty(n_per_seg), n_per_seg = 600; end
    if ~isa(Gd,'tf'), error('Gd debe ser tf discreto.'); end

    % --- Coefs planta (normalizados a a0=1) ---
    [numD, denD] = tfdata(Gd, 'v');
    numD = numD(:).'; denD = denD(:).';
    if abs(denD(1) - 1) > 1e-12
    numD = numD/denD(1);
    denD = denD/denD(1);
    end

    % Asegurar longitud mínima del denominador [1 a1 a2]
    if numel(denD) < 3, denD(end+1:3) = 0; end
    a1 = denD(2);
    a2 = denD(3);

    % b0,b1,b2 sin romper si faltan términos en el numerador
    b0 = 0; b1 = 0; b2 = 0;
    if numel(numD) >= 2, b0 = numD(2); end
    if numel(numD) >= 3, b1 = numD(3); end
    if numel(numD) >= 4, b2 = numD(4); end
    % Nota: esto respeta tu estilo previo (b0=numD(2), b1=numD(3), ...)

    coefs = struct('b0',b0,'b1',b1,'b2',b2,'a1',a1,'a2',a2, ...
    'z0_0',z0_0,'z0_1',z0_1,'p0_0',p0_0,'p0_1',p0_1,'Kc',Kc);

    % --- Tiempo y referencia (refmin -> refmax -> refmin) ---
    N  = 3*n_per_seg;
    td = (0:N-1)' * T;
    refd = [refmin*ones(n_per_seg,1);
    refmax*ones(n_per_seg,1);
    refmin*ones(n_per_seg,1)];

    % --- Inicialización ---
    yd = zeros(N,1);
    ed = zeros(N,1);
    ud = zeros(N,1);

    % --- Precalculos del compensador ---
    P1 = (p0_0 + p0_1);
    P2 = (p0_0 * p0_1);
    Z1 = (z0_0 + z0_1);
    Z2 = (z0_0 * z0_1);

    % --- Loop ---
    for k = 3:N-1
    ed(k) = refd(k) - yd(k);

    % Compensador 2° orden
    u_k = P1*ud(k-1) - P2*ud(k-2) + Kc*( ed(k) - Z1*ed(k-1) + Z2*ed(k-2) );

    % Saturación
    if u_k > umax
    ud(k) = umax;
    elseif u_k < umin
    ud(k) = umin;
    else
    ud(k) = u_k;
    end

    % Planta discreta
    yd(k+1) = b0*ud(k) + b1*ud(k-1) + b2*ud(k-2) - a1*yd(k) - a2*yd(k-1);
    end
    end
```

Listing 2: Funciones desarrolladas para la práctica.

```

    close all
    clear all

    % Cargar CSV
    data = readmatrix('opltab.csv');

    % Ignorar cabeceras, quedarte solo con datos numéricos
    u = data(3:end, 5);  % columna E = salida
    y = data(3:end, 11);  % columna K = entrada
    Ts = 0.000099999997474;  % tiempo de muestreo de B3

    % Crear objeto de identificación
    data_id = iddata(y, u, Ts);

    % Estimar una función de transferencia discreta (ejemplo: 2 polos, 1 cero)
    sys0 = tfest(data_id, 2, 0);

    % % Ver resultado
    %  step(sys);
    % % Crear datos de identificación
    % data_id = iddata(y, u, Ts);
    %
    % % Probar con distintos órdenes
    % sys1 = tfest(data_id, 1, 0);  % 1 polo
    % sys2 = tfest(data_id, 2, 1);  % 2 polos, 1 cero
    %
    % % Comparar
    figure;
    compare(data_id,sys0);

    G = tf(sys0);
    T = 1.25e-3;
    Gd = c2d(G, T, 'zoh');
    disp('Planta digital G(z):')
    zpk(Gd)
    controlSystemDesigner(Gd)

    save('planta.mat', 'G');
```

Listing 3: Modelo mejorado de la planta mediante los datos recolectados del osciloscopio

```

    close all
    clear all

    addpath('..\Lab1\')


    load('planta.mat');
    %% Analizamos en el tiempo
    [tr, ts, wn] = plot_step_info(G);
    disp('Planta continua G(s):')
    G
    zpk(G)

    %% Paso 1: Definir periodo de muestreo
    % Se busca una mejora de 4 en el tiempo de rising , o sea tr = 10 ms

    T = 1.25e-3;
    %Gcl = feedback(G,1);
    %figure;
    %step(Gcl)
    %T = stepinfo(Gcl).SettlingTime/8;
```

```
22
23      %% Paso 2: Digitalizar con ZOH
24      Gd = c2d(G, T, 'zoh');
25      disp('Planta digital G(z):')
26      Gd
27      zpk(Gd)
28      figure;
29      pzmap(Gd);
30      title('Lugar de raíces de G(z).')
31
32      zgrid;
33      z = tf([1 0],1,T);
34
35      %% Paso 3: Análisis en tiempo discreto lazo abierto
36      figure;
37
38      % Obtenemos salida y tiempo de la función step
39      [y, t] = step(Gd);
40
41      % Graficamos con stairs (propio de señales discretas)
42      stairs(t*1000, y, 'LineWidth',1.4);
43
44      title('Respuesta al escalón de la planta digitalizad')
45      xlabel('Tiempo [ms]');
46      ylabel('Salida');
47      grid on;
48      ax = gca;
49      ax.XMinorTick = 'on';   % activamos minor ticks
50      ax.YMinorTick = 'on';
51      grid minor;
52
53      %% Paso 4: Definimimos los valores de la simulación
54      umin = 0;
55      umax =  4.08;
56      refmin = 1.5;
57      refmax = 2.5;
58      n_per_seg = 500;
59      %% Paso 5: Lugar de raíces para zita = 0.7, compensador P
60      %figure
61
62      %rlocus(Gd)
63      %zgrid     % agrega la grilla en el plano-z
64      z0 = 0.380927627742363;
65      p0 = 0.836970379456654;
66      K = 0.226961597782889;
67      C_1 = (z-z0)/(z-p0);
68      figure;
69      Gc1 = Gd*C_1;
70      rlocus(Gc1);
71      title('Lugar de raíces de G(z) compensado con un lag-filter de 1° orden, anulando el polo
        más rápido de la planta.')
72
73      zgrid     % agrega la grilla en el plano-z
74      %pause;
75      %[K, ~] = rlocfind(Gc1);    % hacés click donde querés los polos
76      %K = 0.73139;
77      %K = 0.82543;
78      %K = 1.796463473892606;
79      Gc1f = feedback(K*Gc1,1);
80      %figure;
81      %step(Gc1f)
82      info = stepinfo(Gc1f);
83      info
84
85      info_ext = plot_step_annot(Gc1f, 'la planta compensada con un lag-filter de 1° orden,
        anulando el polo más rápido de la planta.');
86      %step(Gc3f);
```

```
info = stepinfo(Gc1f);
info
info_ext

[td, refd, yd, ud, ed, coefs] = sim_compensador_first_order( ...
Gd, z0, p0, K, T, umin, umax, refmin, refmax, n_per_seg);

Nini = 100;              % descartar al inicio
Nfin = 100;              % descartar al final
idx0 = Nini + 1;         % índice inicial válido
idx1 = length(td) - Nfin;  % índice final válido

td   = td(idx0:idx1)-td(idx0);
refd = refd(idx0:idx1);
yd   = yd(idx0:idx1);
ud   = ud(idx0:idx1);
ed   = ed(idx0:idx1);

% Si querés que el tiempo arranque en 0




figure;

% ---------- Subplot 1 ----------
axAbs = subplot(2,1,1);   % eje izquierdo (absoluto)
hold(axAbs,'on'); grid(axAbs,'on');

% y[k] en rojo (abs)
hY = stairs(axAbs, td, yd, 'r-', 'LineWidth', 1.4);

% Armamos eje derecho transparente
axPct = axes('Position', get(axAbs,'Position'), ...
'Color','none', 'YAxisLocation','right', ...
'XLim', get(axAbs,'XLim'), 'XTick',[], 'Box','off');
hold(axPct,'on');

% r[k] en negro, graficado en %
ref_pct = 100*(refd - refmin)/(refmax - refmin);
hR = stairs(axPct, td, refd, 'k--', 'LineWidth', 1.2);

% ======= Cálculo de límites con margen =======
yd_pct  = 100*(yd - refmin)/(refmax - refmin);
pctAll  = [yd_pct; ref_pct];

% Valores extremos en % con margen del 5 %
rawMin = min(pctAll);
rawMax = max(pctAll);
span   = rawMax - rawMin;
pctMin = rawMin - 0.05*span;
pctMax = rawMax + 0.05*span;

% Redondeamos a múltiplos de 20 para ticks
stepPct = 20;
tickMin = stepPct*floor(pctMin/stepPct);
tickMax = stepPct*ceil (pctMax/stepPct);
pctTicks = tickMin:stepPct:tickMax;

% Convertimos a absolutos
valTicks = refmin + (pctTicks/100)*(refmax - refmin);

% Aplicamos a ambos ejes
set(axAbs,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks);
set(axPct,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks,...
'YTickLabel',compose('%.0f %%',pctTicks));
```

```
154
155     % Etiquetas
156     xlabel(axAbs,'Tiempo [s]');
157     ylabel(axAbs,'Respuesta [valor absoluto]');
158     ylabel(axPct,'Escala relativa a ref [%]');
159     title(axAbs,'Respuesta discreta con compensador 1° orden, anulando el polo más rápido de
        la planta.');
160     legend(axAbs, [hY, hR], {'y[k]', 'r[k]'}, 'Location', 'best');
161
162     % ---------- Subplot 2: esfuerzo de control ----------
163     axU = subplot(2,1,2);
164     stairs(axU, td, ud, 'LineWidth',1.2); grid(axU,'on');
165     yline(axU, umax,'r:'); yline(axU, umin,'r:');
166     xlabel(axU,'Tiempo [s]'); ylabel(axU,'u[k]');
167     title(axU,'Esfuerzo de control con saturación');
168
169     % --- Margen de 5% en eje Y ---
170     uMin = min(ud);
171     uMax = max(ud);
172     span = uMax - uMin;
173     ylim(axU, [uMin - 0.05*span, uMax + 0.05*span]);
174
175
176     grid(axAbs,'on');     % grilla principal
177     grid(axAbs,'minor');  % grilla secundaria
178
179     grid(axPct,'on');
180     grid(axPct,'minor');
181
182     grid(axU,'on');
183     grid(axU,'minor');
184
185
186
187     %% Paso 7: Lugar de raices para zita = 0.7 y y ESS = 0, compensador PI de segundo orden
188     z0_0 = 0.380927627742363;
189     z0_1 = 0.821584532561100;
190
191
192     p0_0 = 0.753973536568200;
193     p0_1 = 1;
194     K = 0.287529112579611;
195     C_3 = ((z-z0_0)*(z-z0_1))/((z-p0_0)*(z-p0_1));
196     Gc3 =Gd*C_3;  %z=0.8433  %8433
197     figure;
198     rlocus(Gc3)
199     title('Lugar de raíces de G(z) compensado con un lag-filter de 2° orden, fijando un polo
        en z = 1 y anulando ambos polos de la planta.');
200
201     zgrid
202     %pause;
203     %[K, ~] = rlocfind(Gc3);    % hacés click donde querés los polos
204     %K = 1.14;
205     %figure;
206     Gc3f= feedback(K*Gc3,1);
207
208     info_ext = plot_step_annot(Gc3f, 'la planta compensada con un lag-filter de 2° orden,
        fijando un polo en z = 1 y anulando ambos polos de la planta.');
209     %step(Gc3f);
210     info = stepinfo(Gc3f);
211     info
212     info_ext
213
214
215     % --- Llamada a tu simulador (misma interfaz que tu first_order) ---
216     [td, refd, yd, ud, ed, coefs] = sim_compensador_second_order( ...
217     Gd, z0_0, z0_1, p0_0, p0_1, K, T, umin, umax, refmin, refmax, n_per_seg);
```

```
218
219
220     Nini = 100;               % descartar al inicio
221     Nfin = 100;               % descartar al final
222     idx0 = Nini + 1;          % índice inicial válido
223     idx1 = length(td) - Nfin;  % índice final válido
224
225     td   = td(idx0:idx1)-td(idx0);
226     refd = refd(idx0:idx1);
227     yd   = yd(idx0:idx1);
228     ud   = ud(idx0:idx1);
229     ed   = ed(idx0:idx1);
230
231
232
233
234     figure;
235
236     % ---------- Subplot 1 ----------
237     axAbs = subplot(2,1,1);   % eje izquierdo (absoluto)
238     hold(axAbs,'on'); grid(axAbs,'on');
239
240     % y[k] en rojo (abs)
241     hY = stairs(axAbs, td, yd, 'r-', 'LineWidth', 1.4);
242
243     % Armamos eje derecho transparente
244     axPct = axes('Position', get(axAbs,'Position'), ...
245     'Color','none', 'YAxisLocation','right', ...
246     'XLim', get(axAbs,'XLim'), 'XTick',[], 'Box','off');
247     hold(axPct,'on');
248
249     % r[k] en negro, graficado en %
250     ref_pct = 100*(refd - refmin)/(refmax - refmin);
251     hR = stairs(axPct, td, refd, 'k--', 'LineWidth', 1.2);
252
253     % ======= Cálculo de límites con margen =======
254     yd_pct  = 100*(yd - refmin)/(refmax - refmin);
255     pctAll  = [yd_pct; ref_pct];
256
257     % Valores extremos en % con margen del 5 %
258     rawMin = min(pctAll);
259     rawMax = max(pctAll);
260     span   = rawMax - rawMin;
261     pctMin = rawMin - 0.05*span;
262     pctMax = rawMax + 0.05*span;
263
264     % Redondeamos a múltiplos de 20 para ticks
265     stepPct = 20;
266     tickMin = stepPct*floor(pctMin/stepPct);
267     tickMax = stepPct*ceil (pctMax/stepPct);
268     pctTicks = tickMin:stepPct:tickMax;
269
270     % Convertimos a absolutos
271     valTicks = refmin + (pctTicks/100)*(refmax - refmin);
272
273     % Aplicamos a ambos ejes
274     set(axAbs,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks);
275     set(axPct,'YLim',[valTicks(1) valTicks(end)],'YTick',valTicks,...
276     'YTickLabel',compose('%.0f %%',pctTicks));
277
278     % Etiquetas
279     xlabel(axAbs,'Tiempo [s]');
280     ylabel(axAbs,'Respuesta [valor absoluto]');
281     ylabel(axPct,'Escala relativa a ref [%]');
282     title(axAbs,'Respuesta discreta con compensador 2° orden, fijando un polo en z = 1 y
        anulando ambos polos de la planta');
283     legend(axAbs, [hY, hR], {'y[k]', 'r[k]'}, 'Location', 'best');
```

```
284
285     % ---------- Subplot 2: esfuerzo de control ----------
286     axU = subplot(2,1,2);
287     stairs(axU, td, ud, 'LineWidth',1.2); grid(axU,'on');
288     yline(axU, umax,'r:'); yline(axU, umin,'r:');
289     xlabel(axU,'Tiempo [s]'); ylabel(axU,'u[k]');
290     title(axU,'Esfuerzo de control con saturación');
291
292     % --- Margen de 5% en eje Y ---
293     uMin = min(ud);
294     uMax = max(ud);
295     span = uMax - uMin;
296     ylim(axU, [uMin - 0.05*span, uMax + 0.05*span]);
297
298
299
300
301
302
303
304     grid(axAbs,'on');     % grilla principal
305     grid(axAbs,'minor');  % grilla secundaria
306
307     grid(axPct,'on');
308     grid(axPct,'minor');
309
310     grid(axU,'on');
311     grid(axU,'minor');
312
313
314     %% impresion antigua
315     % % --- Gráficas estilo informe ---
316     % figure;
317     % subplot(2,1,1);
318     % stairs(td, refd, 'k--','LineWidth',1.0); hold on;
319     % stairs(td, yd,   'LineWidth',1.4);
320     % grid on; xlabel('Tiempo [s]'); ylabel('Respuesta de la planta/Referencia');
321     % title('Respuesta discreta con compensador 2° orden'); legend('r[k]','y[k]','Location','
        best');
322     %
323     % subplot(2,1,2);
324     % stairs(td, ud, 'LineWidth',1.2);
325     % yline(umax,'r:'); yline(umin,'r:');
326     % grid on; xlabel('Tiempo [s]'); ylabel('u[k]');
327     % title('Esfuerzo de control con saturación');
```

Listing 4: Segunda hoja de cálculos utilizada.

APÉNDICE B
CÓDIGO DESARROLLADO PARA EL PSOC

```
1
2       \\XDXDXD
```

Listing 5: Código desarrollado para la implementación de los compensadores con el PSoC en lenguaje C.