Práctica de Laboratorio 4: Diseño de controladores por síntesis directa.

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. Introducción

En este laboratorio se aborda el diseño de un controlador digital para una planta analógica, utilizando la técnica analítica de Truxal–Ragazzini. Este método ofrece un enfoque directo para determinar el controlador que garantiza un comportamiento dinámico deseado en el sistema en lazo cerrado.

El trabajo se centra en tres ejes principales: en primer lugar, se discretiza la función de transferencia de la planta para distintos tiempos de muestreo seleccionados bajo criterios de ingeniería; en segundo lugar, se diseña el controlador digital en dos versiones, una que permite oscilaciones entre muestras y otra de tipo *dead-beat*, que elimina dichas oscilaciones; finalmente, se implementa la ecuación en diferencias del controlador en el PSoC y se contrastan los resultados experimentales con los obtenidos en simulación.

De esta manera, se busca cumplir con los objetivos planteados: diseñar un controlador digital por el método de Truxal–Ragazzini empleando Matlab, implementar su ecuación en diferencias en el PSoC, presentar los resultados obtenidos y realizar una discusión crítica de las diferencias y similitudes respecto a las simulaciones.

II. Objetivos

- Diseño de un controlador digital para una planta analógica por el método de Truxal–Ragazzini utilizando Matlab.
- Implementar la ecuación en diferencias del controlador utilizando el PSoC.
- Presentar los resultados obtenidos y compararlos con los obtenidos por simulación.
- Discutir los resultados.

III. Teoría

Se recomienda consultar la referencia:
M. Fadali, A. Visioli, *Digital Control Engineering – Analysis and Design*, Sección 6.6., donde se desarrollan las bases teóricas necesarias para el diseño de compensadores mediante el diagrama de Bode y la evaluación de estabilidad en sistemas discretos.

1. **Diseño del controlador.** Aplicando el método de Truxal–Ragazzini, se construyen dos versiones del controlador:
   a) *Controlador con oscilaciones entre muestras.* Diseño considerando únicamente el tiempo mínimo de establecimiento.
   b) *Controlador Dead-Beat.* Diseño aplicando las restricciones necesarias para evitar oscilaciones entre muestras.
2. **Simulación y resultados.** Se presentan las respuestas obtenidas para ambos diseños, evaluando métricas de desempeño temporal.
3. **Discusión comparativa.** Se comparan los resultados obtenidos experimentalmente y por simulación, discutiendo las ventajas, limitaciones y diferencias entre ambos enfoques de control.

IV. Obtención de la función de transferencia de la planta.

IV-A. Función de transferencia de la planta 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.

IV-A0a. 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 $V_{cc}/2 = 2.5$ V. En los cálculos posteriores se toma dicho valor como punto de referencia.

IV-A0b. 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):

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

IV-A0c. 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:

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

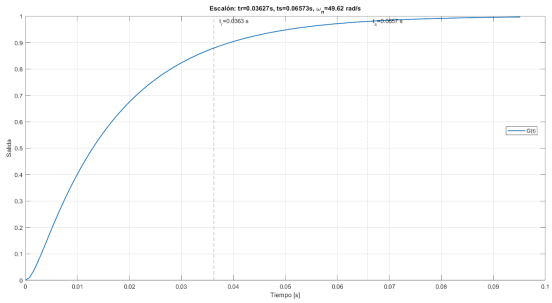


Figura 1. Escalón en lazo abierto del sistema.

IV-B. Discretización de la planta.

La planta a discretizar está dada por:

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

Para discretizarla, se utiliza el método de la *transformación de la respuesta al impulso*. La relación general es:

$$G(z) = (1-z^{-1})\,\mathcal{Z}\left\{\mathcal{L}^{-1}\left(\frac{G(s)}{s}\right)^*\right\}. \tag{3}$$

A partir de la descomposición en fracciones parciales, se obtiene:

$$\frac{G(s)}{s} = \underbrace{\frac{K}{ab}}_{\delta}\cdot\frac{1}{s} + \underbrace{\frac{K}{b^2-ab}}_{\beta}\cdot\frac{1}{s+b} + \underbrace{\frac{K}{a^2-ab}}_{\alpha}\cdot\frac{1}{s+a}, \tag{4}$$

donde los coeficientes α, β y δ corresponden a los términos de la expansión.

Transformada-Z: Aplicando la transformada inversa de Laplace, discretizando la señal con un muestreador de orden cero, y por último aplicando la transformada-Z a cada se obtiene:

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

Finalmente, la planta discretizada puede escribirse como:

$$G(z) = \frac{\gamma z^2 - \theta z + \psi}{(z-e^{-aT})(z-e^{-bT})}, \tag{5}$$

donde los parámetros γ, θ y ψ se expresan en función de los coeficientes de la fracción parcial (4):

$$\gamma = \alpha+\beta+\delta$$
$$\theta = \alpha+\beta+\delta+\alpha e^{-bT}+\beta e^{-aT}+\delta\big(e^{-aT}+e^{-bT}\big)$$
$$\psi = \alpha e^{-bT}+\beta e^{-aT}+\delta e^{-aT}e^{-bT}$$

V. Selección de un tiempo de muestreo adecuado

La selección de los tiempos de muestreo se realizó a partir del criterio de Nyquist $T_n = \frac{2\pi}{w_n}$. Se determinó que el tiempo crítico es $T_n = 139\,\text{ms}$. A partir de este valor, se estableció un margen que permite definir intervalos de muestreo aceptables para el diseño del controlador.

Como punto de partida, se consideró el tiempo de muestreo utilizado en otros laboratorios, $T = 1.25\,\text{ms}$. Con base en ello, se seleccionaron los siguientes valores de muestreo:

$$T = 0.0043\,\text{s}, \quad 0.0087\,\text{s}, \quad 0.0174\,\text{s}, \quad 0.0348\,\text{s}, \quad 0.0695\,\text{s}.$$

VI. Discretización de la planta

A partir de los tiempos de muestreo seleccionados, se obtiene la función de transferencia discreta $G(z)$ para cada caso:

Tiempo de muestreo $T_s = 1.25\,ms$

$$G(z) = \frac{0.02174\,(z+0.7411)}{(z-0.9331)(z-0.4346)}$$

Tiempo de muestreo $T_s = 0.2608\,ms$

$$G(z) = \frac{0.0013411z^2 + 0.0013996z}{0.87989z^2 - 1.8771z + 1}$$

Tiempo de muestreo $T_s = 0.5216\,s$

$$G(z) = \frac{0.004929z^2 + 0.0053679z}{0.7742z^2 - 1.7639z + 1}$$

Tiempo de muestreo $T_s = 1.0433\,s$

$$G(z) = \frac{0.016666z^2 + 0.019765z}{0.59939z^2 - 1.5629z + 1}$$

Tiempo de muestreo $T_s = 2.0866\,s$

$$G(z) = \frac{0.047884z^2 + 0.067322z}{0.35926z^2 - 1.2439z + 1}$$

Tiempo de muestreo $T_s = 4.1731\,s$

$$G(z) = \frac{0.10095z^2 + 0.19895z}{0.12907z^2 - 0.82877z + 1}$$

VII. Diseño del controlador.

VII-A. Con oscilaciones entre muestras.

Se diseñaron los controladores para los distintos tiempos de muestreo seleccionados, con el objetivo de alcanzar un tiempo de establecimiento mínimo. Este diseño parte de la expresión de la función de transferencia en lazo cerrado:

$$G_{cl}(z) = \frac{C(z)\,G_{ZAS}(z)}{1 + C(z)\,G_{ZAS}(z)} \tag{6}$$

Despejando la ecuación (6) en función del controlador $C(z)$, se obtiene:

$$C(z) = \frac{1}{G_{ZAS}(z)} \cdot \frac{G_{cl}(z)}{1 - G_{cl}(z)} \tag{7}$$

En particular, para el diseño de un controlador *con oscilaciones intra-muestra*, simplemente se considera que la función de transferencia en lazo cerrado es $G_{cl}(z) = z^{-1}$. Sustituyendo en (7), la expresión del controlador resulta:

$$C(z) = \frac{1}{G_{ZAS}(z)} \cdot \frac{1}{z - 1} \tag{8}$$

Con los tiempos de muestreo seleccionados, se calculan a continuación los controladores correspondientes con el fin de analizar y comparar su desempeño.

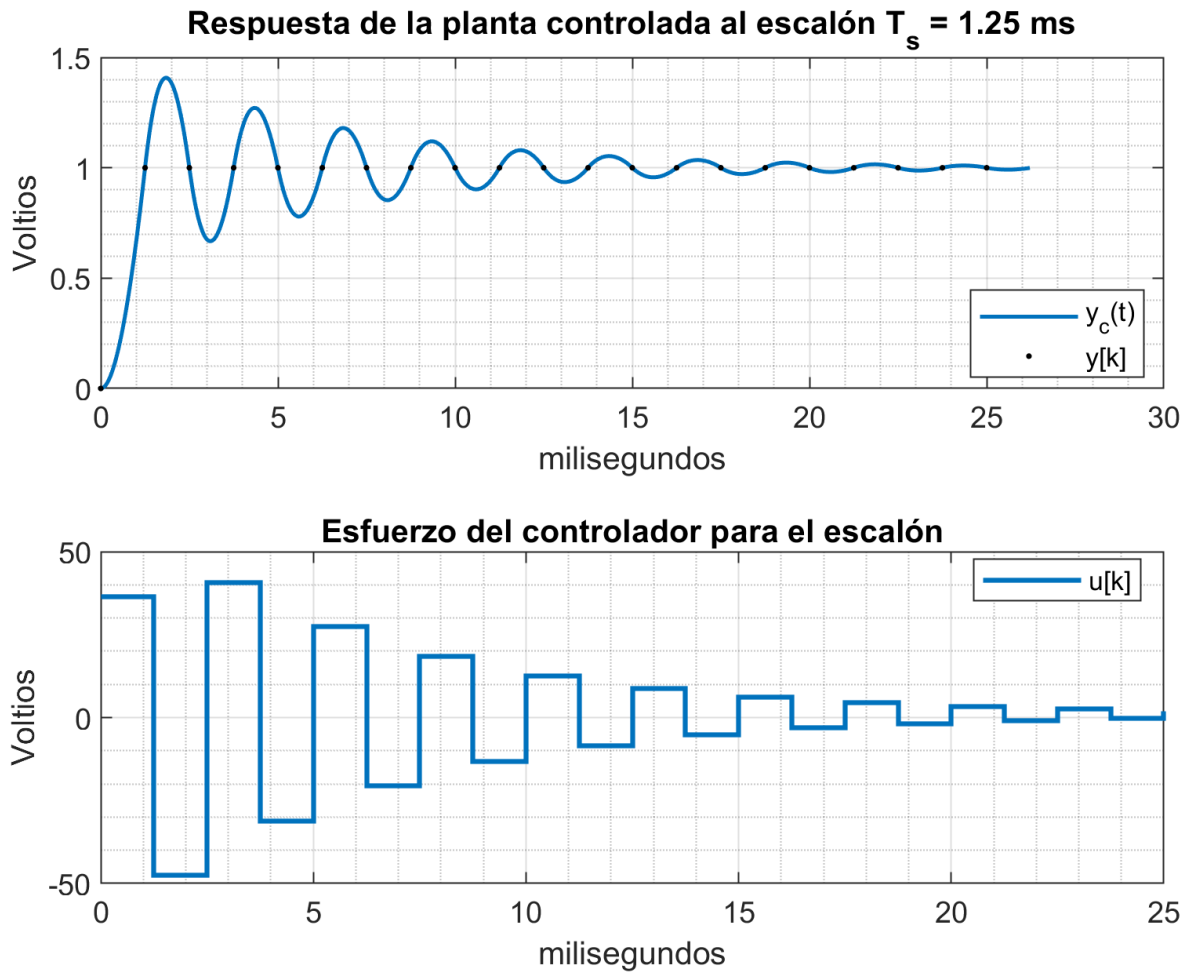


Figura 2. Controlador con tiempo de muestreo $T = 1.25\,ms$.

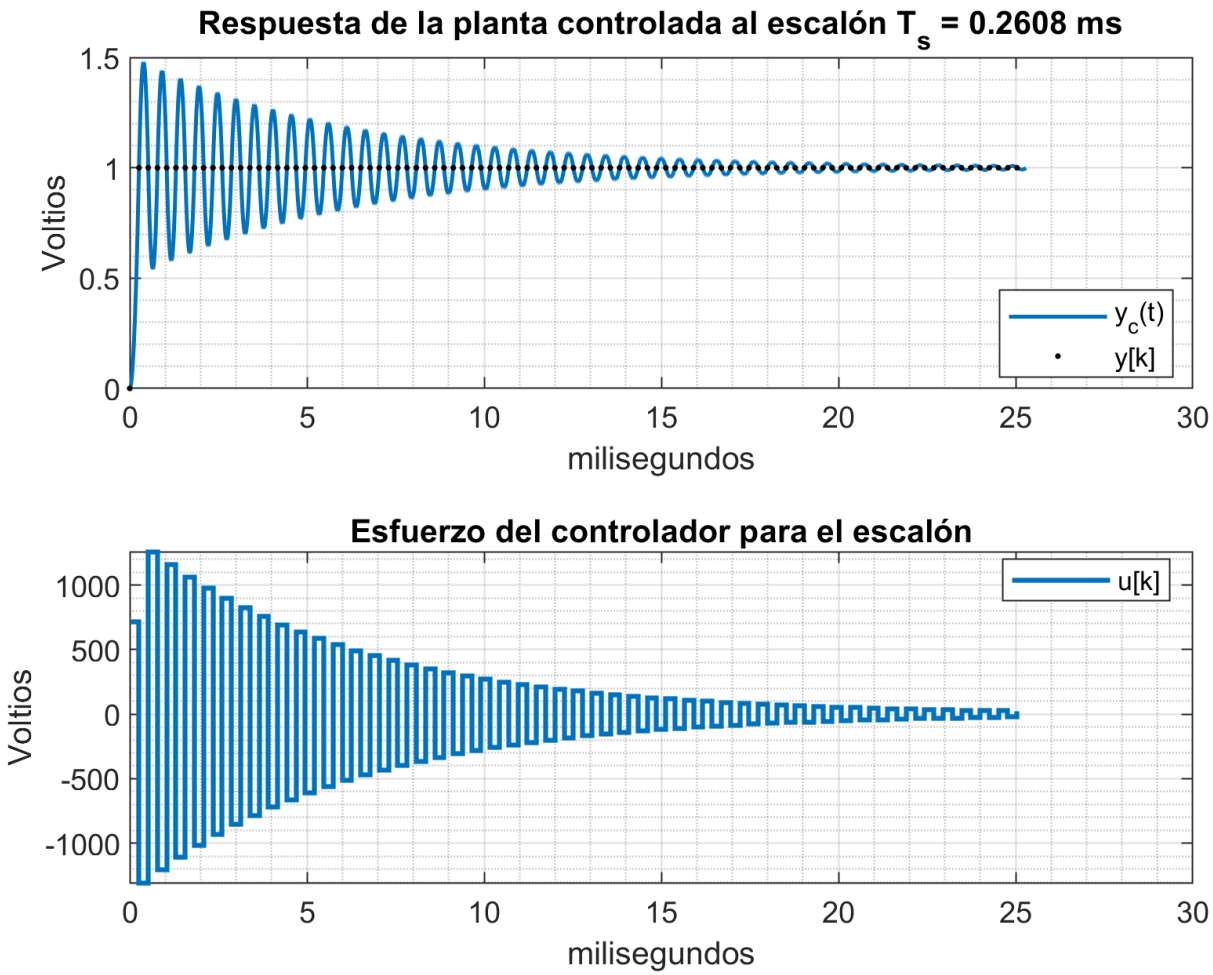


Figura 3. Controlador con tiempo de muestreo $T = 0.2608\,ms$.

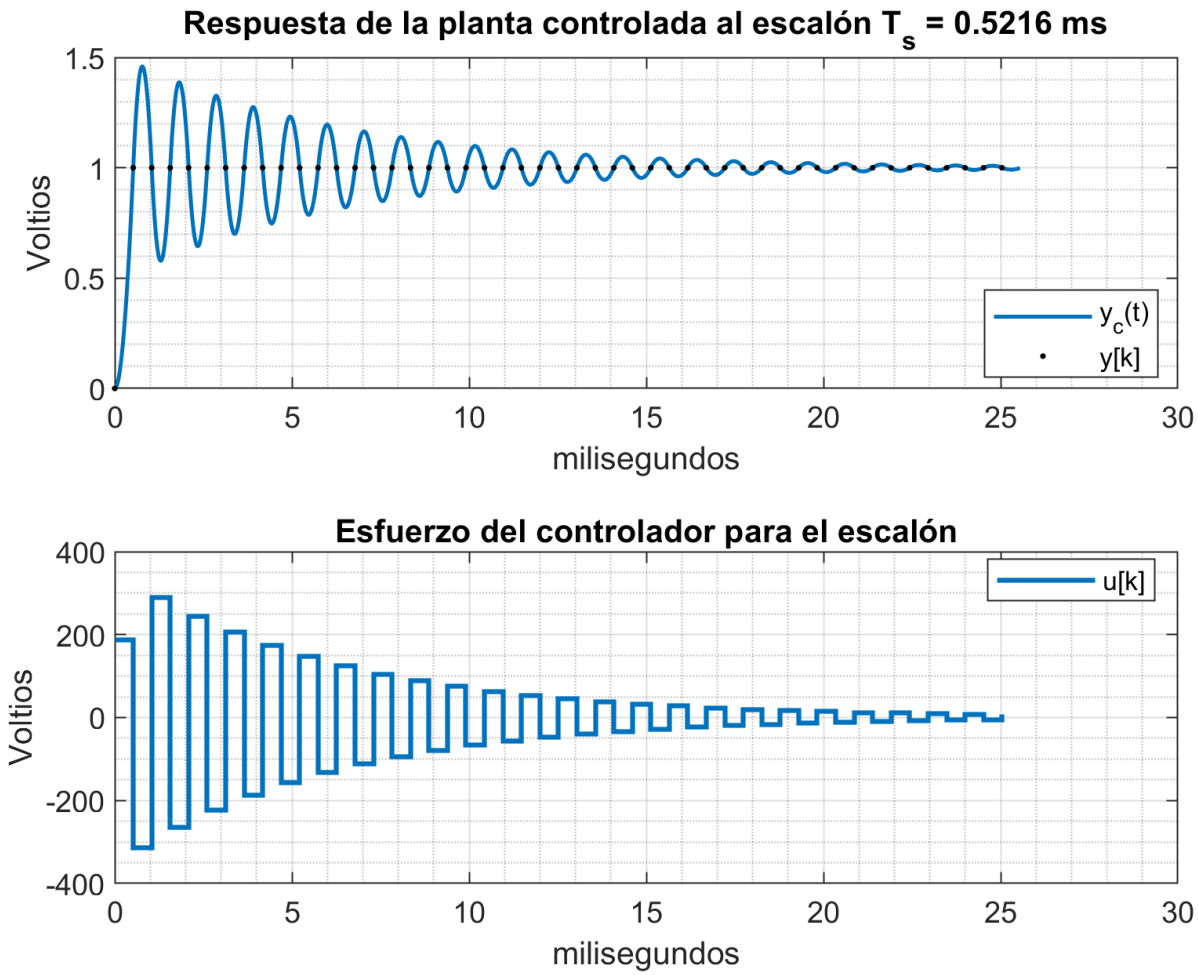


Figura 4. Controlador con tiempo de muestreo $T = 0.5216\,ms$.

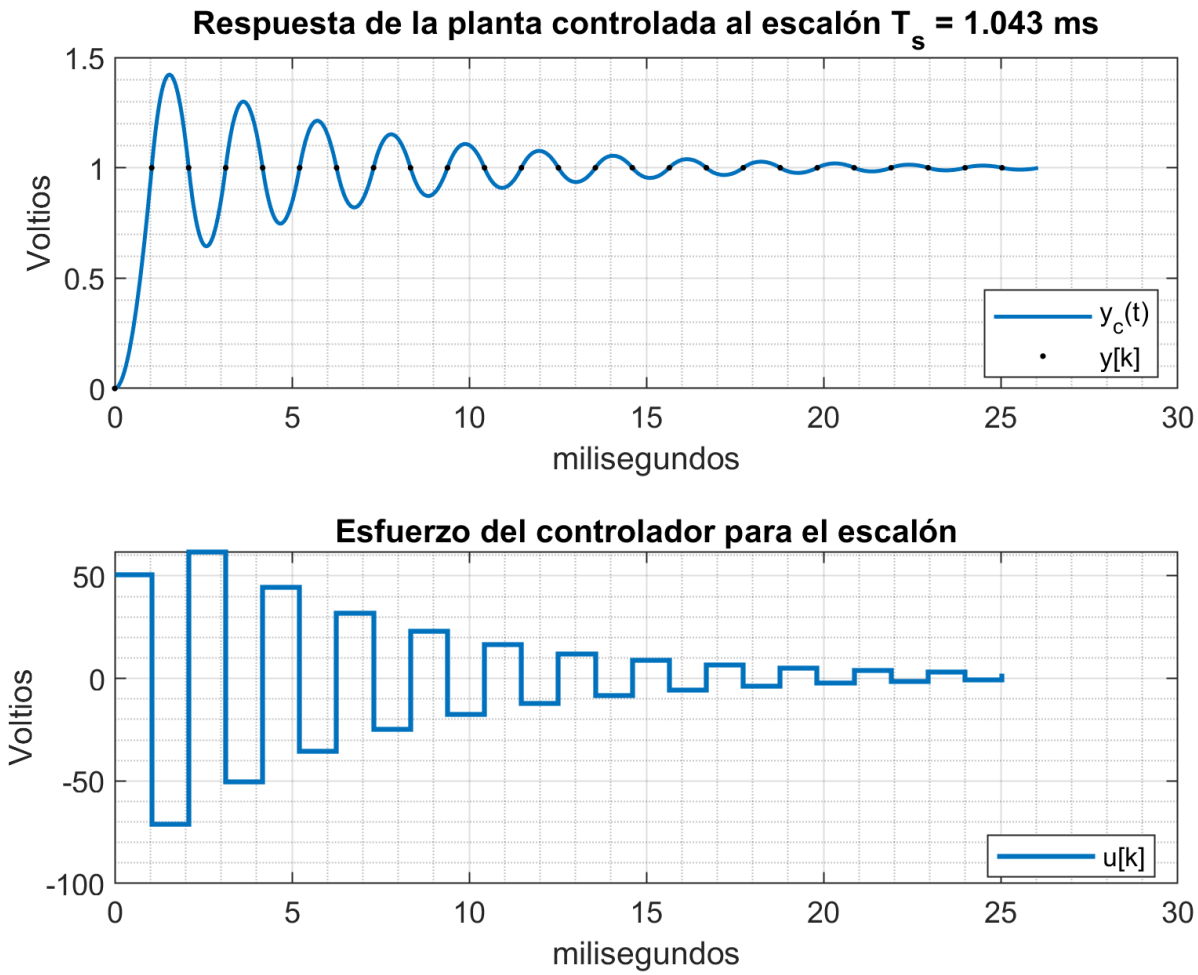


Figura 5. Controlador con tiempo de muestreo $T = 0.1043\,ms$.

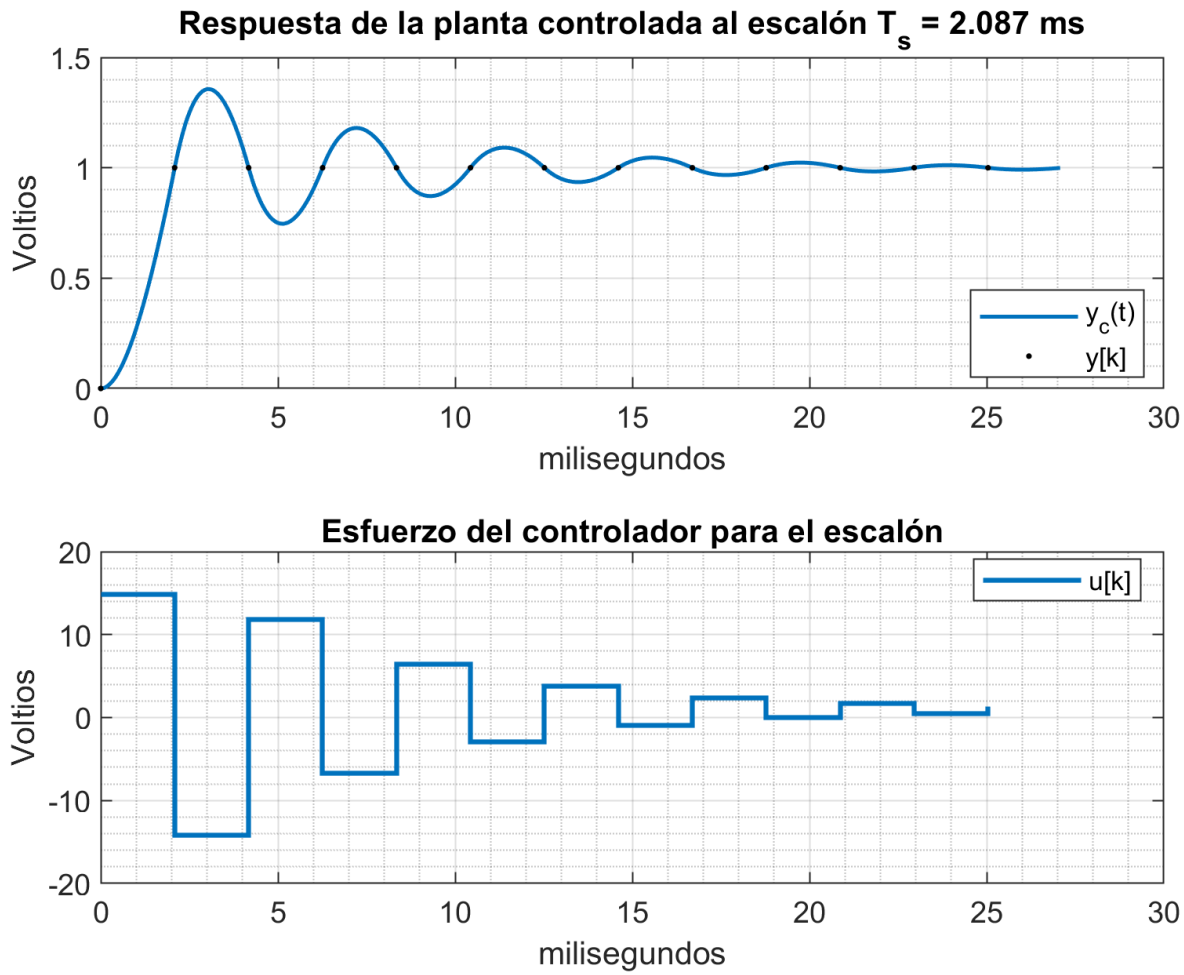


Figura 6. Controlador con tiempo de muestreo $T = 2.087\,ms$.

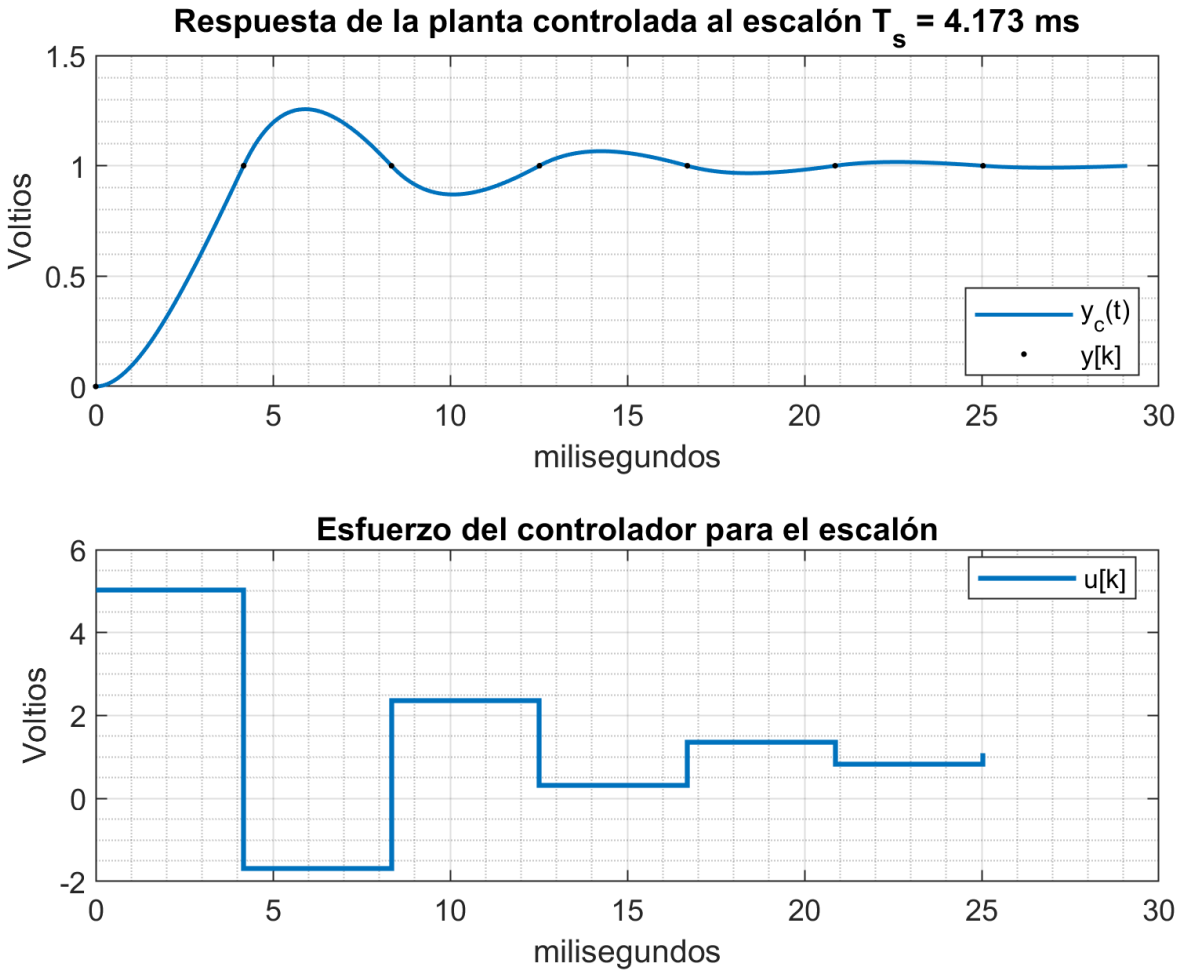


Figura 7. Controlador con tiempo de muestreo $T = 4.173\,s$.

Se puede observar que, a medida que el muestreo es más lento, la respuesta del sistema mejora: la oscilación disminuye y la amplitud del esfuerzo de control se mantiene dentro de un rango aceptable (véase en las Figuras 2 y 7).

VII-B. Eliminando las oscilaciones entre muestras

Para una señal escalón, una respuesta *ripple-free* cumple que, para cualquier $l \geq n$:

- $e(kT) = 0$
- $u(kT) =$ constante

Suponiendo un proceso sin polos ni ceros fuera del círculo unitario y que tampoco presenta un polo en $z = 1$, el controlador puede expresarse como:

$$G_{ZAS}(z) = \frac{a_{n-1}z^{n-1} + \cdots + a_0}{D(z)}$$

De esta forma, el compensador se define como:

$$C(z) = K\,\frac{D(z)}{(z-1)\left(z^{n-1} + b_{n-2}z^{n-2} + \cdots + b_0\right)}$$

Los coeficientes K, $b_{n-1}, \ldots, b_0$ y $a_{n-1}, \ldots, a_0$ están determinados por la siguiente relación:

$$K\left(a_{n-1}z^{n-1} + \cdots + a_0\right) + (z-1)\left(z^{n-1} + \cdots + b_0\right) = z^l, \qquad l \geq n$$

Esta formulación corresponde al diseño de un controlador *deadbeat ripple-free*, el cual garantiza que la señal de control no presente oscilaciones entre muestras y que el error en régimen permanente se anule ante una entrada tipo escalón [1].

Para el caso de la planta analizada (ec. (5)), considerando un sistema de orden dos, se obtiene:

$$K(az+b)+(z-1)(z+b_0)=z^2$$

De donde se determinan los parámetros K y b_0:

$$\begin{cases} b_0 = \dfrac{a_0}{a_0+a_1} = 0.4917 \\ K = \dfrac{1}{a_0+a_1} = 105.0833 \end{cases}$$

Así, el controlador resultante adopta la forma:

$$C(z) = K\,\frac{(z+a)(z+b)}{(z-1)(z+b_0)}$$

Resultados de simulación

A continuación, se presentan los resultados obtenidos para distintos tiempos de muestreo, observándose la evolución del comportamiento del sistema controlado.

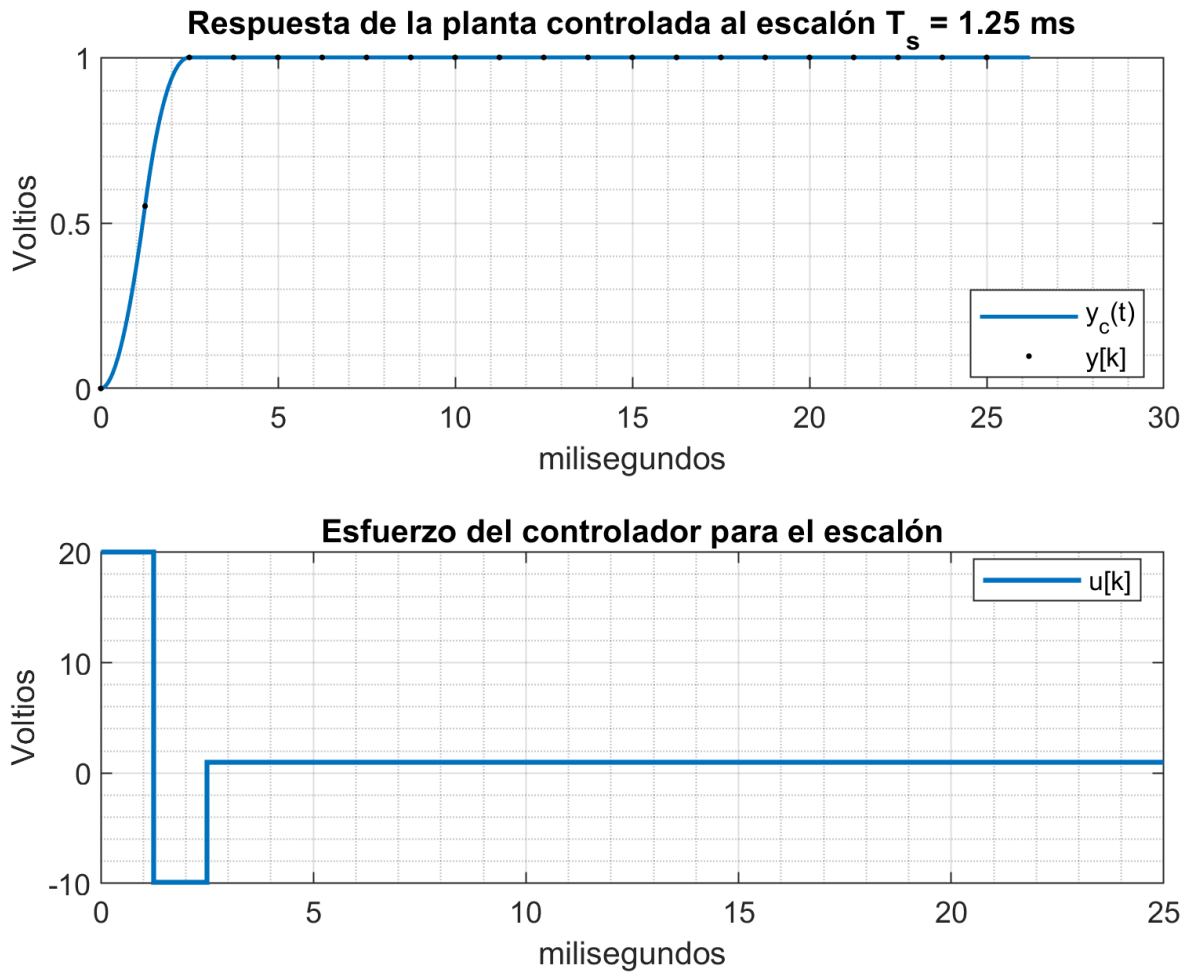


Figura 8. Controlador con tiempo de muestreo $T_s = 1.25\,\text{ms}$.

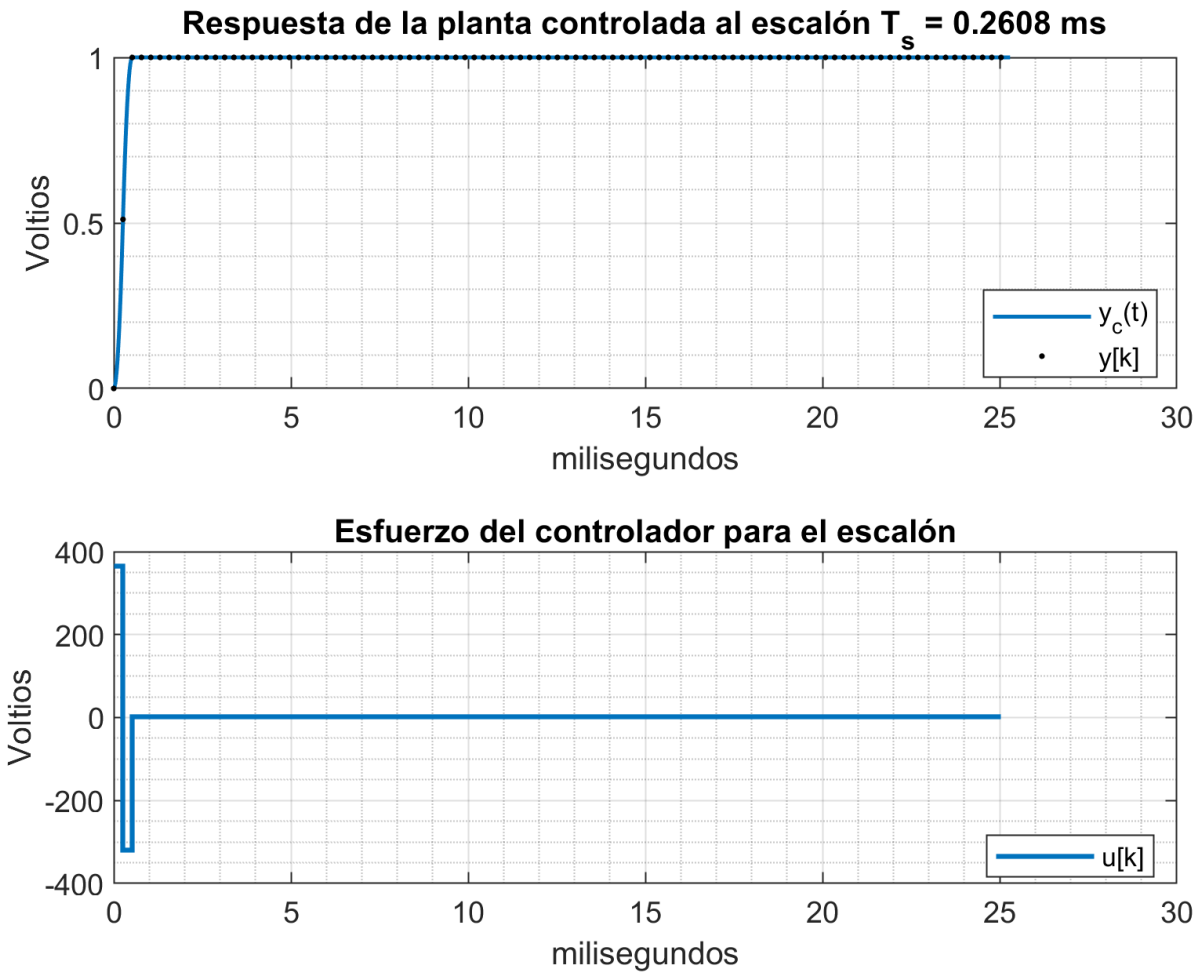


Figura 9. Controlador con tiempo de muestreo $T_s = 0.2608\,\text{ms}$.

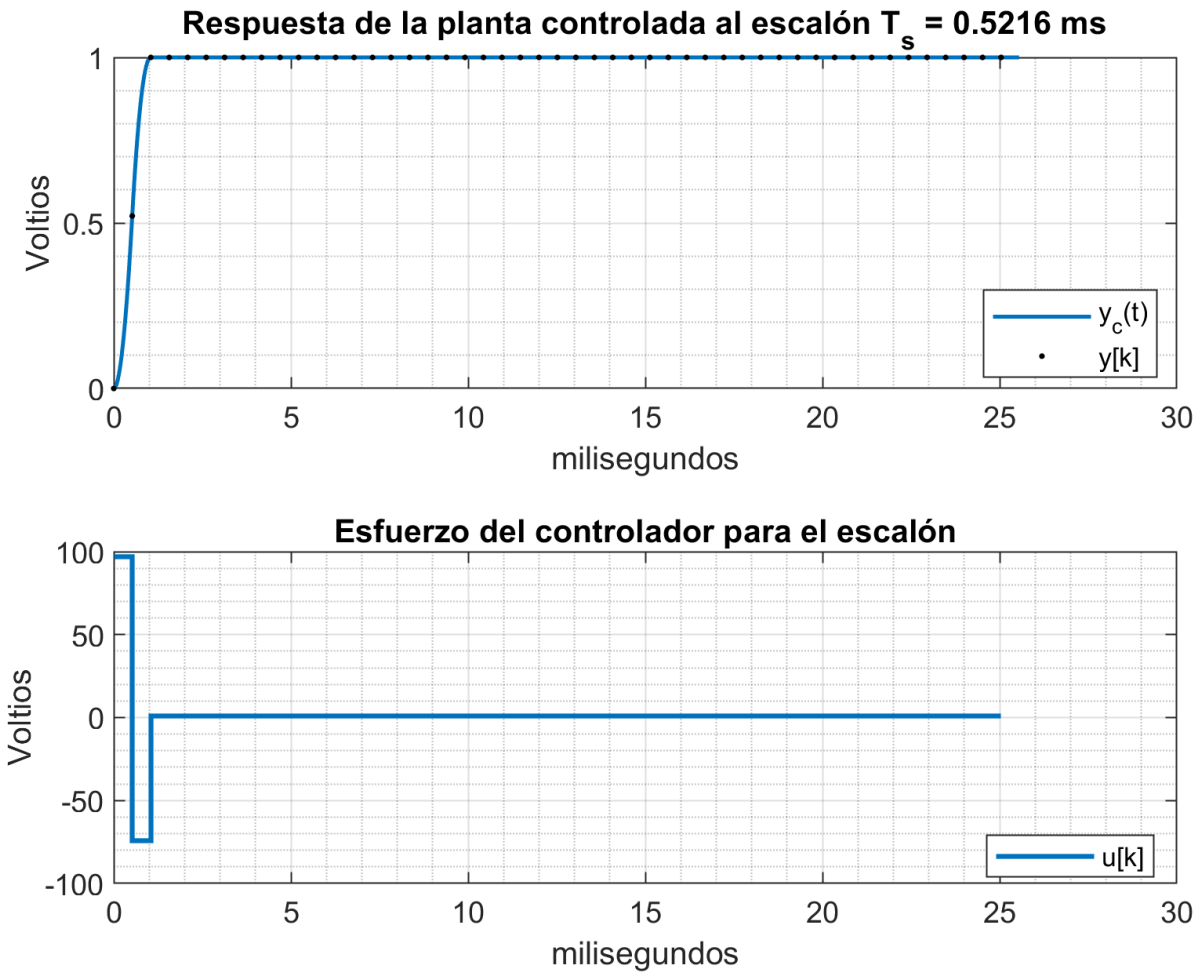


Figura 10. Controlador con tiempo de muestreo $T_s = 0.5216\,\text{ms}$.

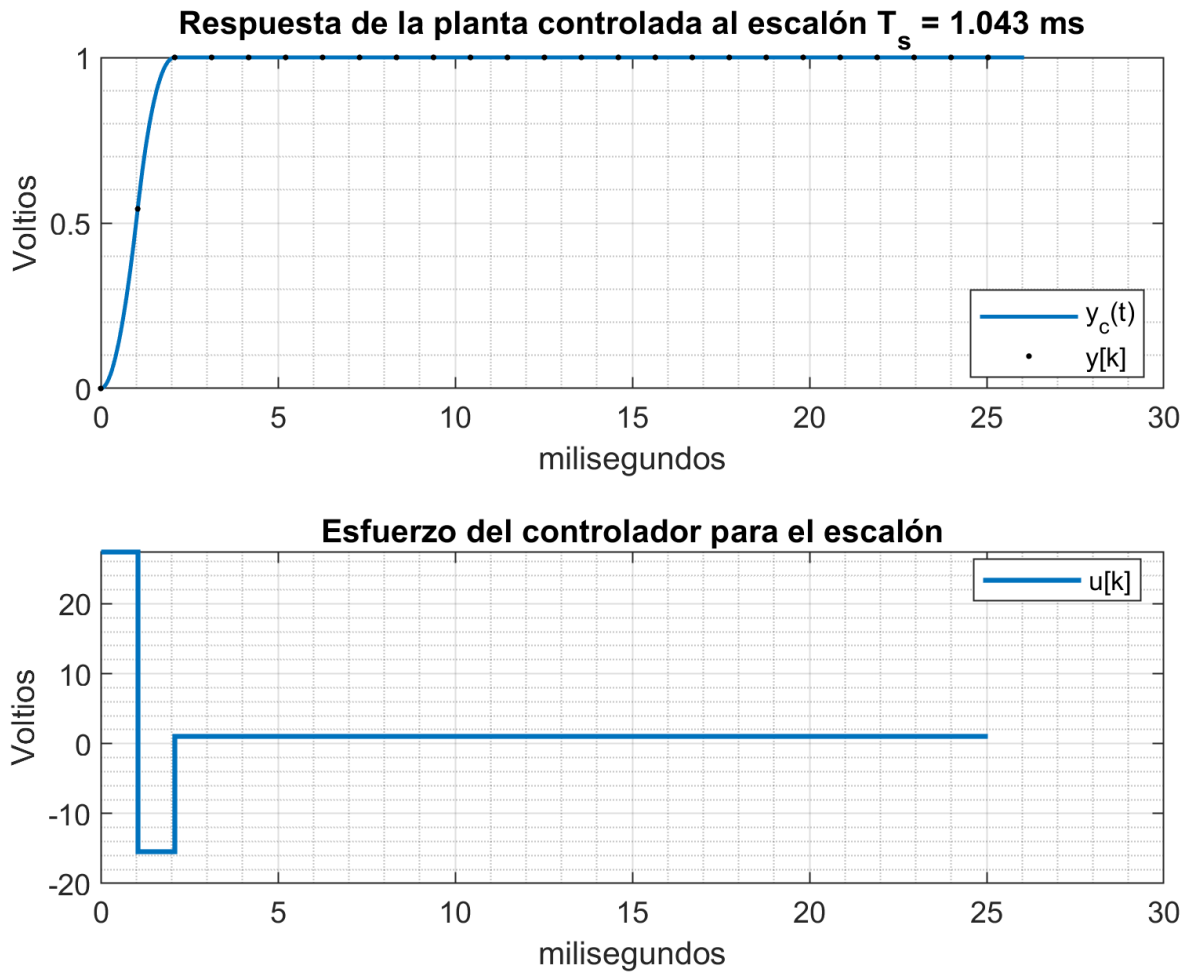


Figura 11. Controlador con tiempo de muestreo $T_s = 1.043\,\text{ms}$.

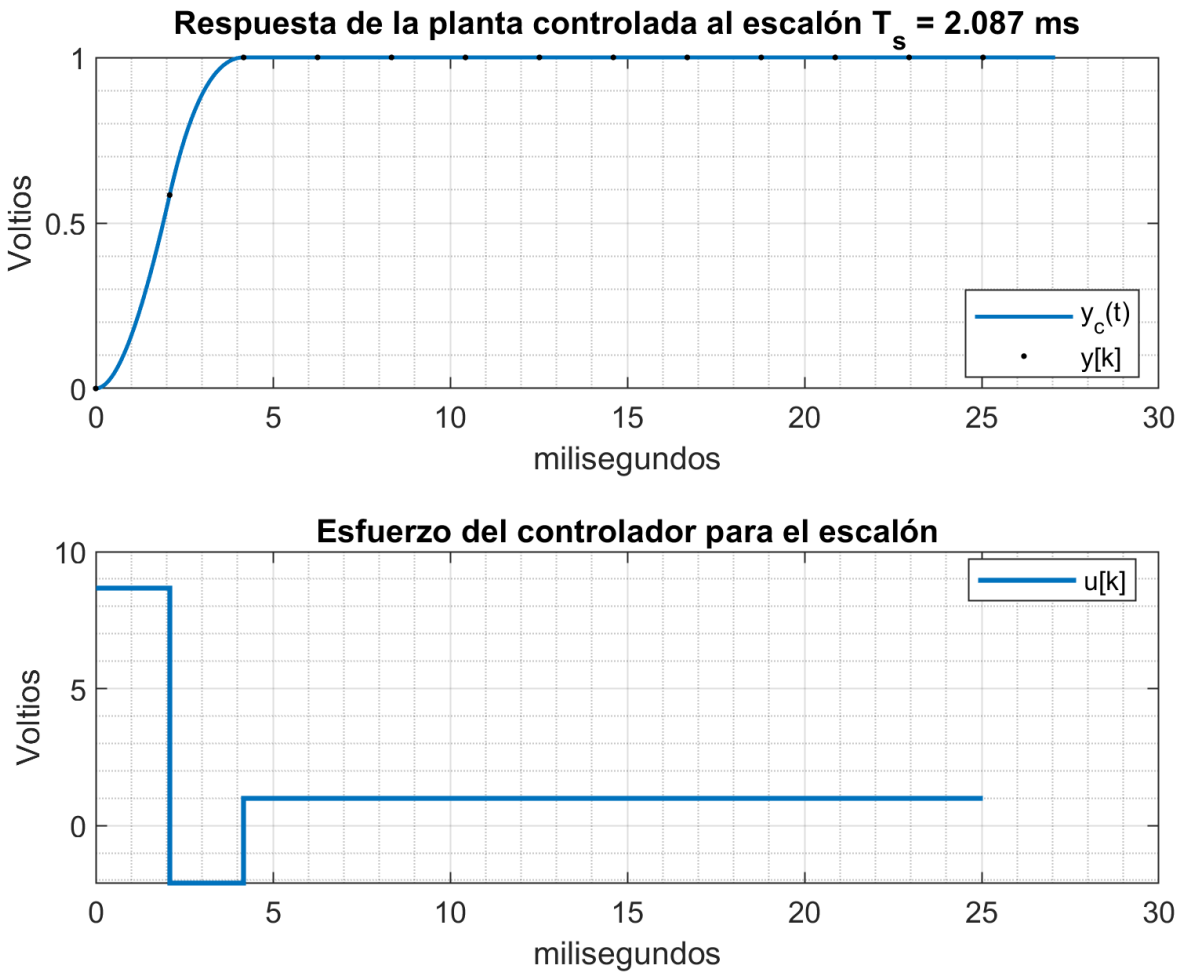


Figura 12. Controlador con tiempo de muestreo $T_s = 2.087\,\text{ms}$.

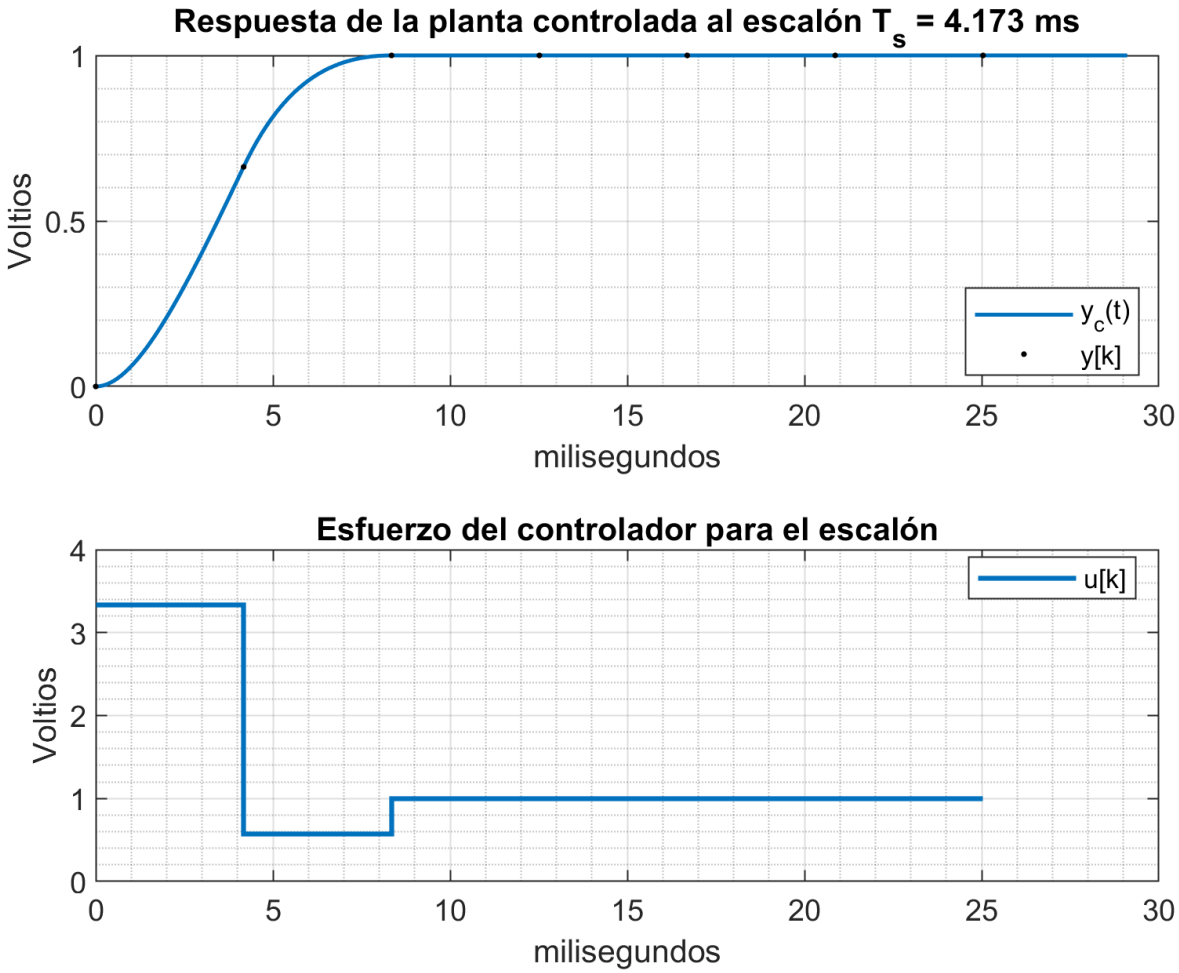


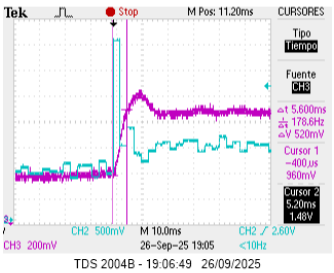
Figura 13. Controlador con tiempo de muestreo $T_s = 4.173\,\text{ms}$.

Se observa una mejora significativa en la reducción de las oscilaciones entre muestras. Sin embargo, a medida que el tiempo de muestreo disminuye, el esfuerzo de control aumenta considerablemente, alcanzando valores que el microcontrolador PSoC no podría manejar de manera estable. Por tanto, para la implementación práctica se selecciona el tiempo de muestreo que permite una respuesta adecuada sin saturar el actuador ni comprometer la estabilidad del sistema.

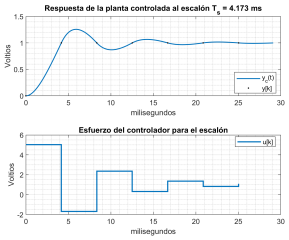
Cuadro I
COMPARACIÓN DE CONTROLADORES DISEÑADOS PARA DISTINTOS TIEMPOS DE MUESTREO

T_s **[s]**	**Planta** $G(z)$	**Compensador método 1** $C_1(z)$	**Compensador método 2** $C_2(z)$
0.00125	$\frac{0.022394z^2 + 0.02747z}{0.54158z^2 - 1.4916z + 1}$	$\frac{0.54158z^2 - 1.4916z + 1}{-0.022394z^2 - 0.0050763z + 0.02747}$	$\frac{10.861z^2 - 29.9141z + 20.0544}{-0.4491z^2 - 0.5509z + 1}$
0.0002608	$\frac{0.0013411z^2 + 0.0013996z}{0.87989z^2 - 1.8771z + 1}$	$\frac{0.87989z^2 - 1.8771z + 1}{-0.0013411z^2 - 5.8441 \times 10^{-5}z + 0.0013996}$	$\frac{321.0458z^2 - 684.9162z + 364.8718}{-0.48934z^2 - 0.51066z + 1}$
0.0005216	$\frac{0.004929z^2 + 0.0053679z}{0.7742z^2 - 1.7639z + 1}$	$\frac{0.7742z^2 - 1.7639z + 1}{-0.004929z^2 - 0.0004389z + 0.0053679}$	$\frac{75.1881z^2 - 171.3039z + 97.1171}{-0.47869z^2 - 0.52131z + 1}$
0.0010433	$\frac{0.016666z^2 + 0.019765z}{0.59939z^2 - 1.5629z + 1}$	$\frac{0.59939z^2 - 1.5629z + 1}{-0.016666z^2 - 0.0030992z + 0.019765}$	$\frac{16.4526z^2 - 42.9003z + 27.449}{-0.45747z^2 - 0.54253z + 1}$
0.0020866	$\frac{0.047884z^2 + 0.067322z}{0.35926z^2 - 1.2439z + 1}$	$\frac{0.35926z^2 - 1.2439z + 1}{-0.047884z^2 - 0.019439z + 0.067322}$	$\frac{3.1184z^2 - 10.7972z + 8.6801}{-0.41564z^2 - 0.58436z + 1}$
0.0041731	$\frac{0.10095z^2 + 0.19895z}{0.12907z^2 - 0.82877z + 1}$	$\frac{0.12907z^2 - 0.82877z + 1}{-0.10095z^2 - 0.097996z + 0.19895}$	$\frac{0.43038z^2 - 2.7635z + 3.3344}{-0.33662z^2 - 0.66338z + 1}$

VIII. Implementación y comparación experimental

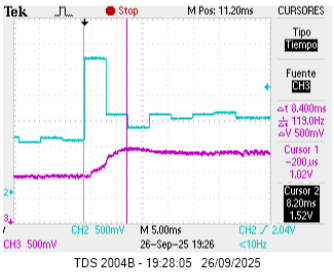


(a) Implementación experimental del método 1.

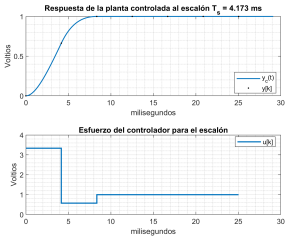


(b) Simulación del método 1 ($T_s = 4.173$ ms).

Figura 14. Comparación entre la simulación y la implementación experimental para el compensador del método 1.



(a) Implementación experimental del método 2.



(b) Simulación del método 2 ($T_s = 4.173$ ms).

Figura 15. Comparación entre la simulación y la implementación experimental para el compensador del método 2.

Para la implementación en el PSoC, se seleccionaron los controladores diseñados que presentaban el menor esfuerzo de control, de modo que su salida fuera físicamente realizable sin saturar el actuador. Si bien los controladores con tiempos de muestreo más pequeños ofrecen una respuesta más rápida, el esfuerzo de control resultante continúa siendo elevado, lo que limita su viabilidad práctica.

Por esta razón, durante las pruebas experimentales se utilizó una referencia de pequeña amplitud, lo que incrementó la relación ruido–señal. Aun así, los resultados fueron satisfactorios, mostrando un seguimiento adecuado y tiempos de establecimiento coherentes con los obtenidos en simulación.

En las Figuras 14 y 15 se presentan las comparaciones entre las simulaciones y los resultados experimentales para ambos métodos de diseño de controladores. Las imágenes experimentales corresponden a una referencia de 1.5 V a 239 Hz, cuyos valores se indican también en el nombre de cada archivo.

Se observa que el primer compensador (método 1) alcanza la referencia en aproximadamente un periodo de muestreo, tal como se había previsto teóricamente, aunque con la presencia de oscilaciones y un leve sobreimpulso, comportamientos también esperados. En cambio, el compensador del método 2 alcanza la estabilidad en torno a los dos periodos de muestreo, presentando una respuesta más suave y sin oscilaciones notorias, aunque con un esfuerzo de control considerablemente mayor y un pequeño sobrepasamiento no observado en las simulaciones.

Al analizar los cursores mostrados en las imágenes experimentales, se verifica que los tiempos de establecimiento concuerdan con los valores estimados en simulación. En ambos casos, la respuesta del sistema puede considerarse satisfactoria, logrando un equilibrio adecuado entre rapidez, estabilidad y esfuerzo de control.

IX. Conclusión

Método	T_{ref} **[ms]**	**Overshoot [%]**	**Oscilaciones**
M1	Sim: 4.173 Med: 5.6 Δ: 34.196 %	Sim: 25 Med: $\approx$ 20 Δ: $\approx$ 20 %	Sim: Sí Med: Sí
M2	Sim: 8.346 Med: 8.4 Δ: 0.647 %	Sim: 0 Med: $<$10 % Δ: $<$10 %	Sim: No Med: No

Cuadro II
Comparación entre simulación y medición para ambos métodos

En síntesis, ambos métodos de diseño mostraron un desempeño adecuado tanto en simulación como en la implementación experimental.

No obstante, este tipo de controladores *analíticos* —tanto el convencional como el *deadbeat*— dependen fuertemente de la precisión del modelo de la planta. En la práctica, los parámetros del sistema pueden variar con la temperatura, el envejecimiento de los componentes o pequeñas no linealidades no contempladas, lo que puede afectar la estabilidad y el desempeño del control.

Si bien los resultados obtenidos validan la teoría en condiciones controladas, su aplicación práctica requiere cierto margen de robustez para adaptarse a variaciones del sistema real.

APÉNDICE A
CÓDIGOS DE MATLAB

A-A. *comparar_intersample.m*

```
function comparar_intersample(S_all, T)
% COMPARAR_INTERSAMPLE - Compara esfuerzos y salidas continuas
% Uso:
%   comparar_intersample(S_all, T)
% Entradas:
%   S_all : cell array de structs devueltos por ver_intersample
%   T     : vector de Ts correspondientes

estilos = {'-','--',':','-.'};  % distintos estilos
colores = lines(numel(T));      % paleta distinta para cada curva
leg = arrayfun(@(Ts) sprintf('T_s = %.4g ms', Ts*1e3), T, 'UniformOutput', false);

figure('Name','Comparativa esfuerzos y respuestas continuas');
tiledlayout(2,1);

% --- Subplot 1: esfuerzos ---
ax1 = nexttile; hold on; grid on; grid minor;
for k = 1:numel(T)
est = estilos{mod(k-1,numel(estilos))+1};
stairs(S_all{k}.tk*1e3, S_all{k}.uk, est, ...
'Color', colores(k,:), 'LineWidth', 1);
end
xlabel('milisegundos');
ylabel('Voltios');
title('Esfuerzos del controlador para distintos T_s');
legend(leg, 'Location', 'best');

% --- Subplot 2: salidas continuas ---
ax2 = nexttile; hold on; grid on; grid minor;
for k = 1:numel(T)
est = estilos{mod(k-1,numel(estilos))+1};
plot(S_all{k}.tc*1e3, S_all{k}.yc, est, ...
'Color', colores(k,:), 'LineWidth', 1);
end
xlabel('milisegundos');
ylabel('Voltio');
title('Respuesta continua de G(s) con u_{ZOH}(t)');
legend(leg, 'Location', 'best');

linkaxes([ax1 ax2],'x');  % sincroniza zoom en X
end
```

Listing 1. Comparación inter-muestra.

A-B. *calculos_auxiliares.m*

```
% Ejemplo simple de simulación con saturación
clear all
close all
% Planta continua: G(s) = 1/(s+1)
s = tf('s');
G = 1/((s+1));

% Controlador discreto PI: C(z) = Kp + Ki*T/(z-1)
Kp = 1; Ki = 0.001; T = 0.01;
C = c2d(pid(Kp,Ki), T, 'tustin');  % discreto

% Parámetros simulación
N = 500;
refd = ones(N,1);       % escalón
umin = -2; umax = 2;    % saturación

```

```
    % Discretizar planta
    Gd = c2d(G,T,'zoh');
    zpk( feedback(C*Gd,1))
    % Obtener coeficientes
    [Nc,Dc] = tfdata(C,'v');  Nc = Nc(:)'; Dc = Dc(:)';
    [Nb,Db] = tfdata(Gd,'v'); Nb = Nb(:)'; Db = Db(:)';

    % Inicializar
    yc = zeros(N,1);    % salida de planta
    u  = zeros(N,1);    % señal aplicada
    uc = zeros(N,1);    % señal de controlador (sin saturación)
    e  = zeros(N,1);

    for k=1:N
    e(k) = refd(k) - yc(k);

    % --- controlador discreto (uc) ---
    uc_k = 0;
    for i=0:length(Nc)-1
    if k-i>=1, uc_k = uc_k + Nc(i+1)*e(k-i); end
    end
    for i=1:length(Dc)-1
    if k-i>=1, uc_k = uc_k - Dc(i+1)*uc(k-i); end
    end
    uc(k) = uc_k;           % señal antes de saturación
    u(k)  = min(max(uc_k,umin), umax);   % saturada

    % --- planta discreta ---
    yk = 0;
    for i=0:length(Nb)-1
    if k-i>=1, yk = yk + Nb(i+1)*u(k-i); end
    end
    for i=1:length(Db)-1
    if k-i>=1, yk = yk - Db(i+1)*yc(k-i); end
    end
    if k<N, yc(k+1)=yk; end
    end

    % Gráficos
    t = (0:N-1)*T;
    subplot(3,1,1); plot(t,yc); ylabel('y[k]')
    subplot(3,1,2); plot(t,u); ylabel('u[k]')
    subplot(3,1,3); plot(t,e); ylabel('e[k]'); xlabel('t [s]')
```

Listing 2. Utilidades auxiliares.

A-C. gen_ref_pulso_blocks.m

```
    function [t, refd] = gen_ref_pulso_blocks(Ts_ref, T_total, ref_amp, pulse_amp, f_hz)
    % Alterna bloques de alto/bajo alrededor de ref_amp con frecuencia f_hz.
    % Alto = ref_amp + pulse_amp
    % Bajo = ref_amp - pulse_amp

    N      = max(1, round(T_total / Ts_ref));      % total de muestras
    t      = (0:N-1)' * Ts_ref;
    Nblk   = max(1, round((1/(2*f_hz)) / Ts_ref)); % muestras por medio periodo

    hi = ref_amp + pulse_amp;
    lo = ref_amp - pulse_amp;

    pattern = [repmat(hi, Nblk, 1); repmat(lo, Nblk, 1)];
    refd = repmat(pattern, ceil(N/(2*Nblk)), 1);
    refd = refd(1:N);
    end
```

Listing 3. Generador de referencia por pulsos (bloques).

A-D. guardar_resultados.m

```
%% ================== Guardar figuras e informe ==================
% Carpeta de salida
outDir = fullfile(pwd, 'resultados');
figDir = fullfile(outDir, 'figuras');
if ~exist(outDir, 'dir'), mkdir(outDir); end
if ~exist(figDir, 'dir'), mkdir(figDir); end

% --- 1) Guardar TODAS las figuras abiertas (las que genera comparar_intersample y otras)
figs = findall(0, 'Type', 'figure');
tsStamp = datestr(now, 'yyyymmdd_HHMMSS');  % timestamp único
for i = 1:numel(figs)
try
set(figs(i), 'PaperPositionMode', 'auto');
fname = sprintf('fig_%02d_%s.png', i, tsStamp);
print(figs(i), fullfile(figDir, fname), '-dpng', '-r300');
catch ME
warning('No pude exportar la figura %d: %s', i, ME.message);
end
end
%%

% --- 2) Armar archivo .txt con: Ts | Planta(G) | C_m1 | C_m2 (solo FT)
txtPath = fullfile(outDir, sprintf('resumen_control_%s.txt', tsStamp));
fid = fopen(txtPath, 'w');
if fid < 0
error('No pude crear el archivo de resumen en %s', txtPath);
end

fprintf(fid, 'Resumen de control por periodo de muestreo\n');
fprintf(fid, 'Formato: Ts [s] | Planta G | C_m1 | C_m2\n');
fprintf(fid, '-----------------------------------------------------------------\n');

% String "limpio" de G continua (única para todas las filas)

Nrows = min([numel(T), numel(C_all_m1), numel(C_all_m2)]);
for k = 1:Nrows
Ts_k   = T(k);
G_show = tf2str_z(Gd_all_m2{k});
C1_show = tf2str_z(C_all_m1{k});
C2_show = tf2str_z(C_all_m2{k});
fprintf(fid, '%.12g | %s | %s | %s\n', Ts_k, G_show, C1_show, C2_show);
end
fclose(fid);

% Guarda también el workspace por si querés reusar
save(fullfile(outDir, sprintf('workspace_%s.mat', tsStamp)), ...
'T','G','C_all_m1','C_all_m2','Gd_all_m1','Gd_all_m2');
fprintf('Listo.\nResumen: %s\n', txtPath);

function s = tf2str_simple(sys)
% Devuelve "(num)/(den)" en una línea.
% - Si es continuo: variable 's'
% - Si es discreto: variable 'z^-1' + sample time
[num, den, Ts] = tfdata(sys, 'v');
if isct(sys)
s = sprintf('(%s)/(%s)', poly2str(num,'s'), poly2str(den,'s'));
else
s = sprintf('(%s)/(%s); Ts=%.10g', poly2str(num,'z^-1'), poly2str(den,'z^-1'), Ts);
end
s = regexprep(s, '\s+', ' ');  % compactar espacios
end
function s = tf2str_z(sys)
% Escribe la FT discreta en variable 'z' (no z^-1)
assert(~isct(sys),'Solo discreto');
[num, den, ~] = tfdata(sys,'v');   % coef. en z^-1 descendentes
```

```
    % Convertimos a polinomios en z multiplicando por z^-N y revirtiendo:
    num_z = fliplr(num); den_z = fliplr(den);
    s = sprintf('(%s)/(%s);', poly2str(num_z,'z'), poly2str(den_z,'z'));
    s = regexprep(s, '\s+', ' ');
    end
```

Listing 4. Guardado de resultados y figuras.

A-E. hojaDeCalculos.m

```
close all
clear all

addpath('..\Lab1\');
addpath('..\Lab2\');
addpath('..\Lab3\');

%% 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)


%% ================== Barrido sobre un vector de T ==================
% Vector de periodos de muestreo a probar (ejemplo: usar T1, T2 y más)
Tn = 2*pi/wn;
T = [1.25e-3 Tn/32 Tn/16 Tn/8 Tn/4 Tn/2];    % <-- ajustá a gusto


% Parámetros de la simulación para ver_intersample
Tsim = Tn;
Nups = 40;    % sobremuestreo del ZOH (submuestras por periodo)

% Resultados de cada Ts
S_all_m1 = cell(1, numel(T));
C_all_m1 = cell(1, numel(T));
Gd_all_m1 = cell(1, numel(T));
for k = 1:numel(T)
Ts = T(k);

% Discretizar la planta con ZOH a Ts
Gd_k = c2d(G, Ts, 'zoh');

% Método 1 (oscilaciones entre muestras): F = z^-1 ; C = F/(Gd*(1-F))
z_k = tf([1 0], 1, Ts);
```

```
    C_k = 1/(Gd_k)*1/(z_k-1);

    % Simulación intersample (usa tu función ver_intersample)
    S_k = ver_intersample(G, C_k, round(Tsim/T(k)), Nups);   %#ok<NASGU>

    % Guardar
    S_all_m1{k}  = S_k;
    C_all_m1{k}  = C_k;
    Gd_all_m1{k} = Gd_k;
    end

    comparar_intersample(S_all_m1, T);


    % %% no funciona
    % % parámetros
    % C = C_all_m1{1};
    % Gd = Gd_all_m1{1};
    % Ts_ref  = C.Ts;      % 0.5 ms entre muestras de referencia
    %
    % % 1) pulso montado
    % f = 1/(Ts*300);
    % Tfin = 3/f;
    % [t, refd] = gen_ref_pulso_blocks(Ts,Tfin,2.05,0.5,f); umin=0; umax=5;
    % figure;
    % plot(t,refd);
    % [yd,ud,ucd,ed,t] = sim_lazo_discreto_sat(Gd,C,refd,umin,umax);
    %
    % figure;
    % subplot(3,1,1); stairs(t,yd); grid on; ylabel('y[k]');hold on;stairs(t,refd)
    % subplot(3,1,2); stairs(t,ud,'-');hold on; stairs( t,ucd,'--'); grid on; ylabel('u[k]');
    legend('u','uc')
    % subplot(3,1,3); stairs(t,ed); grid on; ylabel('e[k]'); xlabel('t [s]')
    %
    %
    % Nups = 40;            % submuestras por periodo (p.ej. 40)
    % Muse = 0;             % 0 => usar todo u[k]
    % S = ver_intersample_desde_u(G, Ts, ud, Muse, Nups);


    %% ================== Metodo 2 (orden 1 del numerador A) ==================
    S_all_m2 = cell(1, numel(T));
    C_all_m2 = cell(1, numel(T));
    Gd_all_m2 = cell(1, numel(T));
    for k = 1:numel(T)
    Ts = T(k);

    % Planta discreta
    Gd_k = c2d(G, Ts, 'zoh');

    % === A(z): orden 1 (A = a1 z + a0). En z^{-1} MATLAB da [a0 a1].
    Azinv = Gd_k.Numerator{1};
    if numel(Azinv) < 2
    error('El numerador de Gd_k no es de orden 1.');
    end
    a0 = Azinv(3);
    a1 = Azinv(2);

    % === K y b0 (¡correctos!)
    denom = a1 + a0;
    if abs(denom) < 1e-12
    error('a1 + a0 = 0 -> no hay solución con l=2 y (z-1)B.');
    end
    K  = 1/denom;
    b0 = a0/denom;

    % === B(z) = z + b0 (en variable z, mayor->menor)
```

```
    Bz = [1, b0];

    % === D(z): quita polos en z=1 del denominador de Gd
    p = zpk(Gd_k).P{1};
    has_pole_at_1 = any(abs(p - 1) < 1e-6);
    p_no1 = p(abs(p - 1) >= 1e-6);
    Dz = poly(p_no1);               % en z (mayor->menor)
    if max(abs(imag(Dz))) < 1e-12, Dz = real(Dz); end

    % === Denominador del controlador según tenga o no polo en z=1
    if has_pole_at_1
    % Caso Eq. (6.54): C(z) = K*D(z) / (z + b0)
    denC_z = Bz;
    else
    % Caso Eq. (6.51): C(z) = K*D(z) / [(z-1)(z + b0)]
    denC_z = conv([1 -1], Bz);
    end

    C_k = tf(K * Dz, denC_z, Ts);

    % Simulación inter-muestra
    S_k = ver_intersample(G, C_k, round(Tsim/Ts), Nups);

    % Guardar
    S_all_m2{k}  = S_k;
    C_all_m2{k}  = C_k;
    Gd_all_m2{k}  = Gd_k;
    end

    comparar_intersample(S_all_m2, T);



    % %% no funciona
    % % parámetros
    % C = C_all_m2{1};
    % Gd = Gd_all_m2{1};
    % Ts_ref  = C.Ts;      % 0.5 ms entre muestras de referencia
    %
    % % 1) pulso montado
    % f = 1/(Ts*300);
    % Tfin = 3/f;
    % [t, refd] = gen_ref_pulso_blocks(Ts,Tfin,2.05,0.5,f); umin=-300; umax=500;
    % figure;
    % plot(t,refd);
    % [yd,ud,ucd,ed,t] = sim_lazo_discreto_sat(Gd,C,refd,umin,umax);
    %
    % figure;
    % subplot(3,1,1); stairs(t,yd); grid on; ylabel('y[k]');hold on;stairs(t,refd)
    % subplot(3,1,2); stairs(t,ud,'-');hold on; stairs( t,ucd,'--'); grid on; ylabel('u[k]');
    legend('u','uc')
    % subplot(3,1,3); stairs(t,ed); grid on; ylabel('e[k]'); xlabel('t [s]')
    %
    %
    % Nups = 40;            % submuestras por periodo (p.ej. 40)
    % Muse = 0;             % 0 => usar todo u[k]
    % S = ver_intersample_desde_u(G, Ts, ud, Muse, Nups);
```

Listing 5. Hoja de cálculos para el diseño de los compensadores.

A-F. sim_lazo_discreto_sat.m

```
    function [yd, ud, ucd, ed, t] = sim_lazo_discreto_sat(Gd, Cd, refd, umin, umax)
    % SIM_LAZO_DISCRETO_SAT
    % Simula un lazo discreto con controlador Cd y planta Gd (ambos tf discretos),
    % separando la señal de control antes de saturación (uc) y la aplicada (u).
    %
```

```
    % Entradas:
    %   Gd   : planta discreta (tf) en z, causal
    %   Cd   : controlador discreto (tf) en z, causal
    %   refd : referencia discreta r[k] (vector fila o columna)
    %   umin, umax : límites de saturación del esfuerzo
    %
    % Salidas:
    %   yd  : salida de la planta y[k] (mismo largo que refd)
    %   ud  : esfuerzo aplicado u[k] (saturado)
    %   ucd : esfuerzo del controlador sin saturar u_c[k]
    %   ed  : error e[k] = r[k] - y[k]
    %   t   : tiempo discreto (k*Ts)

    Ts = 0; try, Ts = Cd.Ts; catch, end
    if isempty(Ts) || Ts <= 0, error('Cd debe tener Ts > 0.'); end

    % ===== Coeficientes en z^{-1} y normalización a(0)=1 =====
    [Nc, Dc] = tfdata(Cd, 'v');  Nc = Nc(:).'; Dc = Dc(:).';
    [Nb, Db] = tfdata(Gd, 'v');  Nb = Nb(:).'; Db = Db(:).';

    if abs(Dc(1)) < 1e-12, error('Controlador impropio/no causal: Dc(1)=0'); end
    if abs(Db(1)) < 1e-12, error('Planta discreta impropia/no causal: Db(1)=0'); end

    % Normalizar para que los denominadores empiecen en 1
    Nc = Nc / Dc(1);   Dc = Dc / Dc(1);
    Nb = Nb / Db(1);   Db = Db / Db(1);

    % ===== Preparar señales =====
    refd = refd(:);            % columna
    N = numel(refd);
    t = (0:N-1).' * Ts;

    % y con un paso extra para causalidad explícita (y[k+1])
    y  = zeros(N+1,1);
    ed = zeros(N,1);
    ucd = zeros(N,1);    % señal del controlador (sin saturación)
    ud  = zeros(N,1);    % señal aplicada (saturada)

    % ===== Bucle causal =====
    % Ecuaciones:
    %   Dc(q^-1) * uc[k] = Nc(q^-1) * e[k]
    %   Db(q^-1) * y[k]  = Nb(q^-1) * u[k]
    %
    % Ojo: acá actualizamos y(k+1) para respetar el retardo de ZOH.

    for k = 1:N
    % error con la salida disponible del mismo índice (y[k])
    ed(k) = refd(k) - y(k);

    % ---- Controlador (sin saturación): uc[k] ----
    uc_k = 0.0;

    % término por e[k-i]
    for i = 0:length(Nc)-1
    if k-i >= 1
    uc_k = uc_k + Nc(i+1) * ed(k-i);
    end
    end

    % realimentación por uc[k-i], i >= 1
    for i = 1:length(Dc)-1
    if k-i >= 1
    uc_k = uc_k - Dc(i+1) * ucd(k-i);
    end
    end

    ucd(k) = uc_k;
```

```
73
74      % ---- Saturación y esfuerzo aplicado ----
75      ud(k) = min(max(uc_k, umin), umax);
76
77      % ---- Planta: y[k+1] con u[k-i] e y[k-i] ----
78      y_next = 0.0;
79
80      % parte directa por u[k-i]
81      for i = 0:length(Nb)-1
82      if k-i >= 1
83      y_next = y_next + Nb(i+1) * ud(k-i);
84      end
85      end
86
87      % realimentación por y[k-i], i >= 1
88      for i = 1:length(Db)-1
89      if k-i >= 1
90      y_next = y_next - Db(i+1) * y(k-i+1);   % nota: y es (k+1)
91      end
92      end
93
94      % avanzar estado de la planta
95      y(k+1) = y_next;
96      end
97
98      % Salida alineada con refd
99      yd = y(1:N);
100     end
```

Listing 6. Simulación de lazo discreto con saturación.

A-G. ver_intersample.m

```
1       function S = ver_intersample(G, C, M, N)
2       % VER_INTERSAMPLE - Visualiza oscilaciones entre-muestras con ZOH explícito
3       % Uso:
4       %   S = ver_intersample(G, C, M, N)
5       % Entradas:
6       %   G : planta continua (tf/ss continuo)
7       %   C : controlador digital (tf/ss discreto, con Ts > 0)
8       %   M : 0 => simular hasta t = SettlingTime(G)
9       %       >0 => simular M*Ts
10      %   N : factor de sobremuestreo del ZOH (submuestras por periodo, ej. 30)
11
12      Ts = getTsStrict(C);
13      if Ts <= 0
14      error('El controlador C debe ser discreto con Ts > 0.');
15      end
16
17      % Discretizar la planta
18      Gd = c2d(G, Ts, 'zoh');
19
20      % Lazos cerrados
21      Gcl = feedback(C*Gd, 1);   % r->y[k]
22      Ucl = feedback(C, Gd);     % r->u[k]
23
24      Tfinal = M*Ts;
25
26      K = max(1, floor(Tfinal/Ts));
27      tk = (0:K).' * Ts;
28
29      % Respuestas discretas
30      [yk, ~] = step(Gcl, tk);
31      rk = ones(size(yk));
32      ek = rk - yk;
33      uk = lsim(C, ek, tk);
34
```

```
    % ZOH explícito
    [tc, uc] = zoh_stretch(uk, Ts, N);
    yc = lsim(G, uc, tc);

    % ====== SUBPLOTS (x en ms) ======
    figure('Name','Intersample analysis');
    tiledlayout(2,1);

    % y_c(t) y y[k]
    nexttile;
    plot(tc*1e3, yc, 'LineWidth',1.2); grid on; hold on; grid minor;
    stairs(tk*1e3, yk, 'k.');
    xlabel('milisegundos'); ylabel('Voltios');
    title(sprintf('Respuesta de la planta controlada al escalón T_s = %.4g ms',Ts*1e3));
    legend('y_c(t)','y[k]','Location','best');

    % u[k]
    nexttile;
    stairs(tk*1e3, uk, 'LineWidth',1.5); grid on; grid minor;
    xlabel('milisegundos'); ylabel('Voltios');
    title('Esfuerzo del controlador para el escalón');
    legend('u[k]','Location','best');

    % Salida estructurada
    S = struct('Ts', Ts, 'Tfinal', tk(end), ...
    'tk', tk, 'yk', yk, 'uk', uk, 'ek', ek, ...
    'tc', tc, 'uc', uc, 'yc', yc, ...
    'Gd', Gd, 'Gcl', Gcl, 'Ucl', Ucl);
    end

    % Helpers
    function Ts = getTsStrict(sys)
    Ts = 0;
    try, Ts = sys.Ts; catch, end
    if isempty(Ts), Ts = 0; end
    end

    function [t_hi, u_hi] = zoh_stretch(u_k, T, M)
    if size(u_k,2) > 1, u_k = u_k(:); end
    u_hi = repelem(u_k, M);
    t_hi = (0:numel(u_hi)-1).' * (T/M);
    end
```

Listing 7. Visualización del comportamiento inter-muestra.

A-H. *ver_intersample_desde_u.m*

```

    function S = ver_intersample_desde_u(G, Ts, ud, M, N)
    % VER_INTERSAMPLE_DESDE_U - Simula inter-muestra con G(s) y un esfuerzo u[k] dado.
    % Uso:
    %   S = ver_intersample_desde_u(G, Ts, ud, M, N)
    % Entradas:
    %   G  : planta continua (tf/ss continuo)
    %   Ts : periodo de muestreo del esfuerzo u[k]
    %   ud : vector de esfuerzo discreto u[k] (fila o columna)
    %   M  : (opcional) n° de muestras a usar
    %        0 o [] => usar todo ud
    %        >0     => usar las primeras M muestras
    %   N  : (opcional) sobremuestreo del ZOH (submuestras por periodo, ej. 30; default 30)
    %
    % Salida (struct):
    %   S.Ts, S.tk, S.uk, S.yk       : señales discretas (k*Ts)
    %   S.tc, S.uc, S.yc             : señales continuas (ZOH + lsim(G))
    %   (y figura con y_c(t) y y[k], y u[k])

    if nargin < 5 || isempty(N), N = 30; end
```

```
    if nargin < 4 || isempty(M), M = 0;  end
    if Ts <= 0, error('Ts debe ser > 0.'); end

    % --- Acomodar u[k] y recorte opcional ---
    uk = ud(:);                         % columna
    if M > 0
    M = min(M, numel(uk));
    uk = uk(1:M);
    end

    K  = numel(uk) - 1;                 % última muestra = k=K
    tk = (0:K).' * Ts;

    % --- Reconstrucción inter-muestra por ZOH explícito ---
    [tc, uc] = zoh_stretch(uk, Ts, N);     % uc(t) pieza-constante
    yc       = lsim(G, uc, tc);            % salida continua

    % Muestras de la salida continua exactamente en k*Ts
    yk = yc(1:N:end);    % cada N submuestras corresponde a un instante k*Ts
    % (Por construcción, length(yk) == length(uk) == length(tk))

    % --- Plots (eje en ms) ---
    figure('Name','Intersample desde u[k]');
    tiledlayout(2,1);

    % y_c(t) y y[k]
    nexttile; hold on; grid on; grid minor;
    plot(tc*1e3, yc, 'LineWidth', 1.2);
    stairs(tk*1e3, yk, 'k.', 'LineWidth', 1.0);
    xlabel('t [ms]'); ylabel('y');
    title(sprintf('Salida continua y_c(t) y muestras y[k]  (T_s = %.4g ms)', Ts*1e3));
    legend('y_c(t)','y[k]','Location','best');

    % u[k]
    nexttile; hold on; grid on; grid minor;
    stairs(tk*1e3, uk, 'LineWidth', 1.2);
    xlabel('t [ms]'); ylabel('u[k]');
    title('Esfuerzo aplicado');

    % --- Salida estructurada ---
    S = struct('Ts', Ts, ...
    'tk', tk, 'uk', uk, 'yk', yk, ...
    'tc', tc, 'uc', uc, 'yc', yc);
    end

    % ===== Helper: ZOH explícito =====
    function [t_hi, u_hi] = zoh_stretch(u_k, T, M)
    % Repite cada muestra M veces (ZOH) y arma el vector de tiempo continuo
    if size(u_k,2) > 1, u_k = u_k(:); end
    u_hi = repelem(u_k, M);
    t_hi = (0:numel(u_hi)-1).' * (T/M);
    end
```

Listing 8. .]Reconstrucción inter-muestra desde u[k].

REFERENCIAS

[1] M. S. Fadali and A. Visioli, *Digital Control Engineering: Analysis and Design*, 3rd ed. Academic Press, 2020.