Práctica de Laboratorio 7: Sistemas de seguimiento — Control integral en el espacio de estados

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 el laboratorio anterior se trabajó con un sistema en lazo abierto, donde no existía corrección automática ante perturbaciones ni derivas en los componentes. Esto hacía al sistema más sensible y dependiente de la exactitud del modelo.

En esta práctica se implementa un esquema en **lazo cerrado** mediante realimentación de estados, incorporando además una **acción integral** para asegurar seguimiento de referencia con error estacionario nulo. El integrador acumula el error entre la salida y la referencia, mientras que la realimentación permite ajustar la respuesta dinámica de manera precisa mediante la ubicación arbitraria de polos.

De esta forma se obtiene un controlador más robusto y con mejor comportamiento frente a perturbaciones, combinando la teoría de control en el espacio de estados con la implementación digital en PSoC.

II. Objetivos

- Diseñar un controlador digital en lazo cerrado con realimentación de estados y acción integral.
- Implementar el control en la planta analógica utilizando el PSoC.
- Comparar el desempeño simulado y experimental, analizando la influencia del integrador y la robustez frente a perturbaciones.

III. Cálculos y metodología de diseño

En esta sección se describe, a nivel conceptual y paso a paso, el flujo de diseño implementado en el script de `MATLAB`. El código completo se incluye en el **Anexo**.

III-A. Modelo continuo y parámetros físicos

A partir de los valores medidos de resistencias y capacitores se construye el modelo continuo en espacio de estados. Se definen las constantes de tiempo y ganancias:

$$\tau_1 = R_2 C_1, \qquad \tau_2 = R_4 C_2, \qquad k_1 = -\frac{R_2}{R_1}, \qquad k_2 = -\frac{R_4}{R_3}.$$

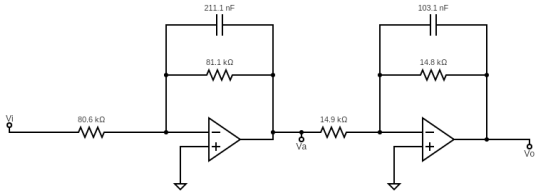


Figura 1. Esquema conceptual de la planta y medición de salida $y = Cx$.

Con ello, las matrices del sistema continuo quedan:

$$F = \begin{bmatrix} -\frac{1}{\tau_1} & 0 \\ \frac{k_2}{\tau_2} & -\frac{1}{\tau_2} \end{bmatrix}, \qquad G = \begin{bmatrix} \frac{k_1}{\tau_1} \\ 0 \end{bmatrix}, \qquad H = \begin{bmatrix} 0 & 1 \end{bmatrix}, \qquad J = 0.$$

Se crea $\mathrm{sysC} = \mathrm{ss}(F, G, H, J)$ y su tf asociada para consulta rápida.

III-B. Selección de T_s y discretización ZOH

Para fijar un período de muestreo informado por la dinámica, se estima la frecuencia del polo más rápido del sistema continuo:

$$f_n \approx \frac{\max |\Re\{p_i(G(s))\}|}{\pi}, \qquad T_n = \frac{1}{f_n},$$

y se toma

$$T_s = \frac{T_n}{4},$$

lo que da al menos ocho muestras sobre la constante de tiempo dominante. Con este T_s se discretiza por ZOH:

$$(A, B, C, D) = \mathrm{ssdata}\big(\mathrm{c2d}(\mathrm{sysC}, T_s, \texttt{'zoh'})\big).$$

III-C. Realimentación de estados (sin integrador)

Se elige un par de polos en $\mathbb{C}$ para el lazo cerrado sin integrador

$$p_{\text{ctrl}} = \{0{,}8 \pm j\,0{,}2\},$$

y se calcula la ganancia de estado K por asignación de polos:

$$K = \text{acker}(A, B, p_{\text{ctrl}}).$$

Para seguimiento de referencia con ganancia unitaria se usa un prefiltro N_{bar} (vía rutina `refi`):

$$u_k = N_{\text{bar}}\, r_k - K\, \hat{x}_k.$$

III-D. Estimadores de estado: predictivo y actual

Se fijan polos rápidos para el estimador

$$p_{\text{obs}} = \{0{,}2 \pm j\, 0{,}2\},$$

y se calculan las ganancias:

$$L_{\text{pred}} = \text{acker}(A^\top, C^\top, p_{\text{obs}})^\top, \qquad L_{\text{act}} = \text{acker}(A^\top, (CA)^\top, p_{\text{obs}}$$

Predictivo: actualiza con la salida en k:

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L_{\text{pred}}\big(y_k - C\hat{x}_k\big).$$

Actual: usa la predicción intermedia $z_k = A\hat{x}_k + Bu_k$ y la salida en $k{+}1$:

$$\hat{x}_{k+1} = z_k + L_{\text{act}}\big(y_{k+1} - Cz_k\big).$$

III-E. Acción integral en espacio de estados (sistema aumentado)

Para eliminar error estacionario frente a referencias/perturbaciones constantes se introduce un estado integral v_k del error $e_k = r_k - y_k$:

$$v_{k+1} = v_k + e_k = v_k + (r_k - Cx_k).$$

Se define el sistema *aumentado* (Ogata 6.7; Franklin 8.5):

$$A_{\text{aug}} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix}, \quad B_{\text{aug}} = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad C_{\text{aug}} = \begin{bmatrix} C & 0 \end{bmatrix},$$

y se asignan polos $\mathcal{P}_i = \{p_{\text{ctrl}},\, p_i\}$ con un polo real adicional p_i para el integrador (p. ej. 0,6). Se obtiene

$$K_{\text{aug}} = \text{acker}(A_{\text{aug}}, B_{\text{aug}}, \mathcal{P}_i) \;\Rightarrow\; K_{\text{aug}} = \begin{bmatrix} K_2 & -K_1 \end{bmatrix},$$

donde K_2 actúa sobre x y K_1 sobre v. En el código, para mantener la ley $u_k = K_1 v_k - K_2 \hat{x}_k$ coherente con el prefiltro, se reexpresa K_{aug} mediante el sistema auxiliar

$$\underbrace{\begin{bmatrix} A - I & B \\ CA & CB \end{bmatrix}}_{\text{Aux}}$$

y la relación (Ogata 6.19):

$$K_2 K_1 = \big(K_{\text{aug}} + [\,0 \;\; I\,]\big)\, \text{Aux}^{-1},$$

desde donde se particiona K_2 (componentes sobre x) y K_1 (componente integral).

Los observadores para el caso con integrador reutilizan el mismo conjunto rápido:

$$L_{\text{pred},i},\; L_{\text{act},i} \quad \text{con} \quad p_{\text{obs},i} = \{0{,}2 \pm j\, 0{,}2\}.$$

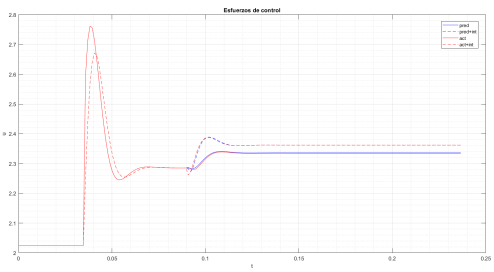


Figura 2. Lazo cerrado con estado integral v y realimentación $K = [K_2 \; -K_1]$. Se muestran los esfuerzos de control en los distintos esquemas.

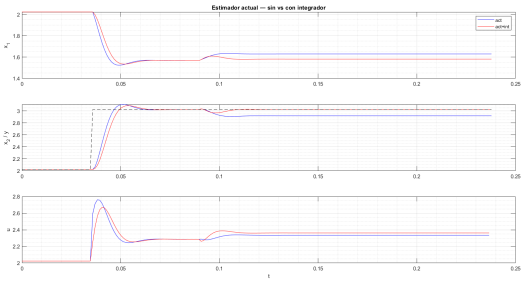


Figura 3. Respuesta al escalón con estimador **actual**. Con ruido en la planta, el integrador corrige la deriva en la salida, a diferencia del caso sin integral.

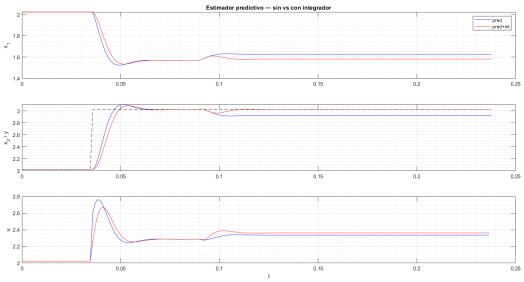


Figura 4. Respuesta al escalón con estimador **predictivo**. Se observa el mismo efecto: el integrador elimina el error estacionario frente a perturbaciones.

III-F. Escenarios de simulación

Se simulan cuatro configuraciones: (1) estimador **predictivo** sin integrador; (2) estimador **actual** sin integrador; (3) **predictivo + integrador**; (4) **actual + integrador**. Se usa un escalón r de amplitud unitaria (con retardo inicial para ver el transitorio) y un ruido de medición w activado en la segunda mitad de la simulación para contrastar sensibilidad. Se acumula un *offset* de visualización (off $= 2{,}024$) para superponer las curvas en escala física.

- **Leyes de control**:

  sin int: $\quad u_k = N_{\text{bar}}\, r_k - K\, \hat{x}_k,$

  con int: $\quad u_k = K_1\, v_{k+1} - K_2\, \hat{x}_k, \quad v_{k+1} = v_k + (r_k - y_k).$

- **Observadores**: actualización *predictiva* con y_k y *actual* con y_{k+1} vía z_k.

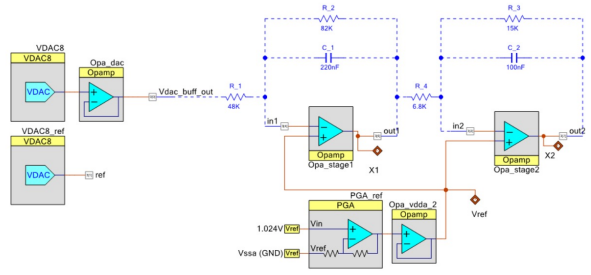


Figura 7. Diagrama del circuito de la planta utilizado en el montaje experimental.

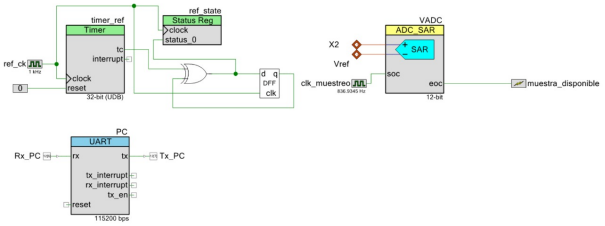


Figura 8. Implementación del controlador **predictivo** con integrador en PSoC (diagrama de control).

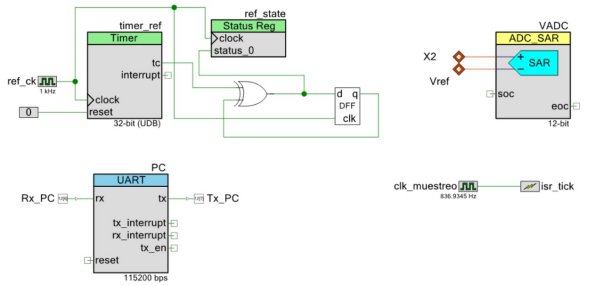


Figura 9. Implementación del controlador **actual** con integrador en PSoC (diagrama de control).

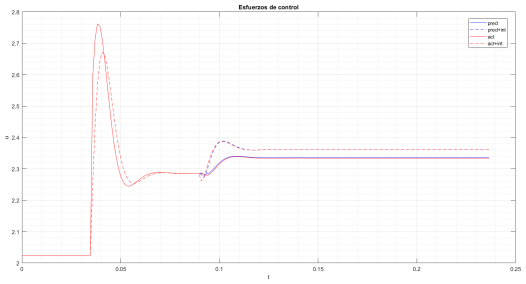


Figura 5. Esfuerzos de control para las cuatro configuraciones (sin/con integrador; estimador actual/predictor).

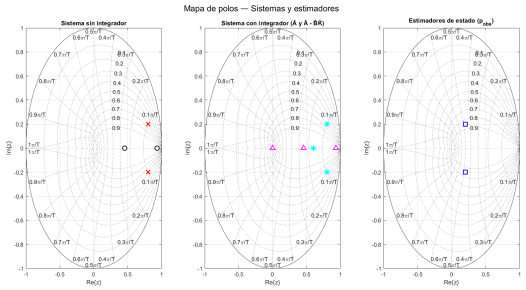


Figura 6. Polos en el plano Z: incluye (i) planta sin integrador antes/después, (ii) planta con integrador antes/después y (iii) estimador de estados (actual y predictor).

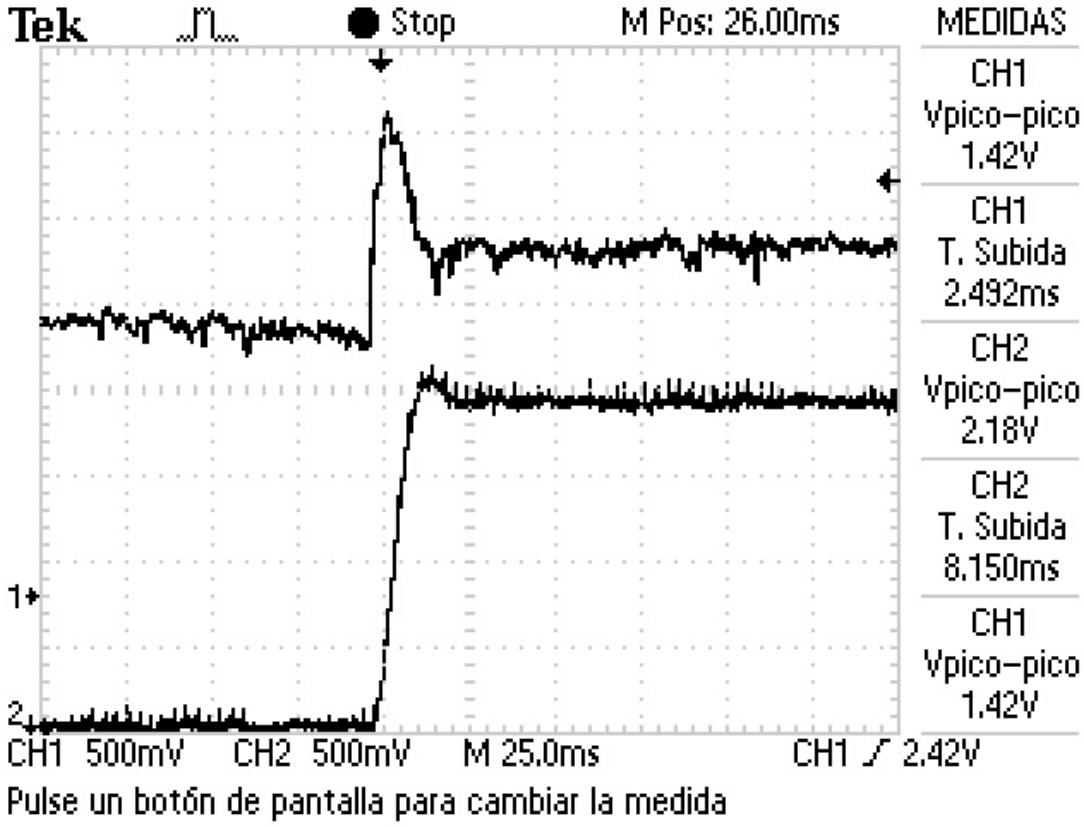
El script completo de `MATLAB` *utilizado para obtener* K, N_{bar}, L, K_1, K_2 *y para generar las simulaciones y mapas de polos se incluye íntegramente en el* ***Anexo***.

IV. Implementación

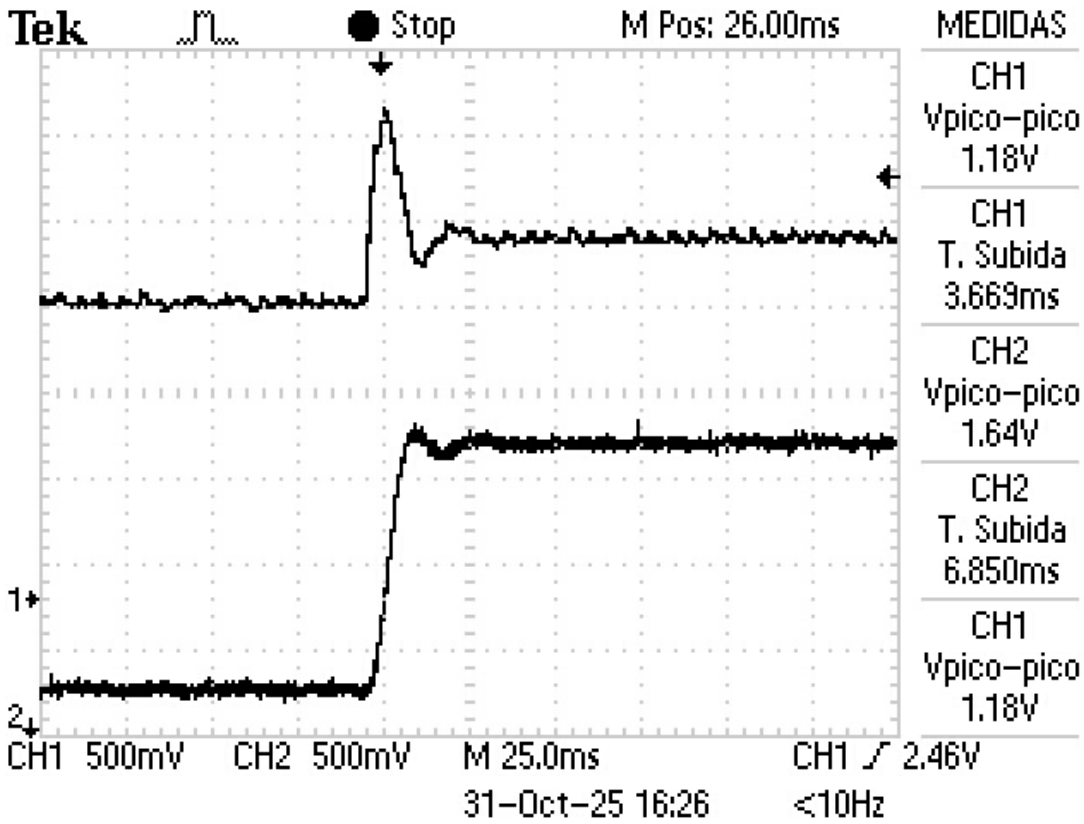
En este laboratorio se reutilizaron los proyectos del Lab 6 (*predictivo* y *actual*) manteniendo la misma estructura de hardware en el PSoC. La única modificación fue la incorporación del integrador en el código C, lo que permitió que ambos esquemas trabajaran ahora en lazo cerrado con acción integral.

$$v_{k+1} = v_k + (r_k - y_k), \qquad u_k = K_1 v_{k+1} - K_2 \hat{x}_k.$$

Los bloques físicos (planta, DAC, ADC, UART e ISR de adquisición) se mantuvieron idénticos al laboratorio anterior, por lo que la adaptación fue directa y sin cambios en el diagrama del sistema.

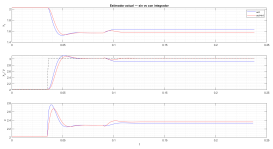


(a) Respuesta experimental con estimador **predictivo** (con integrador).

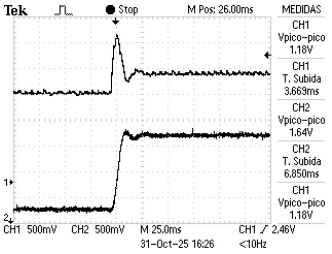


(b) Respuesta experimental con estimador **actual** (con integrador).

Figura 10. Capturas del osciloscopio: ambos esquemas con acción integral muestran eliminación del error estacionario y seguimiento.

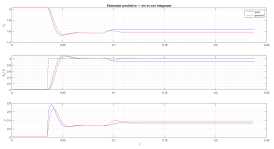


(a) Estimador **actual** (sim.): *sin* vs *con* integrador.

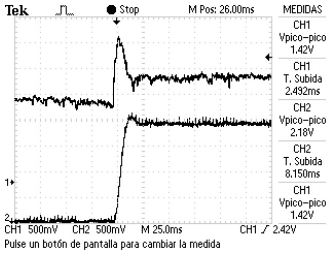


(b) Estimador **actual** (exp.): *con* integrador.

Figura 12. Comparación del estimador actual: simulación *sin/con* integrador vs resultado experimental *con* integrador.



(a) Estimador **predictivo** (sim.): *sin* vs *con* integrador.



(b) Estimador **predictivo** (exp.): *con* integrador.

Figura 11. Comparación del estimador predictivo: simulación *sin/con* integrador vs resultado experimental *con* integrador.

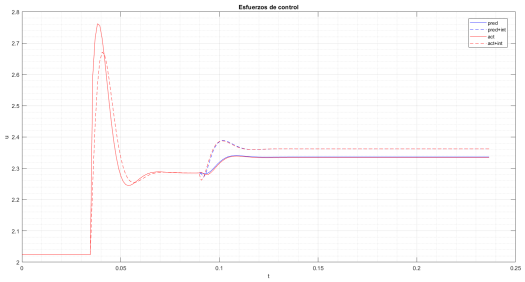


Figura 13. Esfuerzos de control para las cuatro configuraciones: *sin* y *con* integrador, estimador *predictivo* y *actual* (simulación).

Cuadro I
COMPARACIÓN EXPERIMENTAL Y SIMULADA: *sin integrador* / *con integrador*.

Caso	t_s [ms] (*sin*/*con*)	OS [%] (*sin*/*con*)
Predictor (Sim)	7.00/8.00	4.6/4.0
Actual (Sim)	7.00/8.00	4.6/4.0
Predictor (Exp)	10.0/4.0	6.520/8.150
Actual (Exp)	5.940/6.850	10.0/5.0

V. RESULTADOS

En las figuras se observa que la incorporación de la acción integral elimina la deriva frente a perturbaciones y garantiza error estacionario nulo, manteniendo una dinámica global comparable a la simulada. Tanto el observador *predictivo* como el *actual* replican la tendencia teórica, aunque el sobreimpulso real resulta no se asemeja a lo calculado para todos los casos. Las discrepancias pueden atribuirse a tolerancias de componentes pasivos, desajustes de ganancia y cuantización DAC/ADC, además de pequeños desvíos de polos respecto a los ubicados por diseño.

La tabla I resume los valores simulados y experimentales para ambos estimadores, contrastando *sin* y *con* integrador. Se confirma que el integrador elimina el error estacionario y mejora la robustez, a costa de un leve aumento del esfuerzo de control.

APÉNDICE

En este anexo se incluyen los archivos utilizados para el diseño y la implementación: (i) la hoja de cálculos y simulación en `MATLAB`, y (ii) los dos `main.c` del PSoC para los esquemas *predictivo* y *actual*. El código completo se referencia a continuación.

A. *Hoja de cálculos y simulaciones (`MATLAB`)*

```
%% ====================== DISEÑO CON POLOS OBJETIVO ÚNICOS ======================
clear all; clc; close all;
addpath('..\');

%% --- Planta continua (tu modelo) ---
C2 = 103.07e-9;  C1 = 211.1e-9;
R3 = 6.74e3;      R4 = 14.760e3;
R1 = 46.4e3;       R2 = 81.09e3;

tau1 = R2*C1;            k1 = -R2/R1;
tau2 = R4*C2;            k2 = -R4/R3;

F = [ -1/tau1,     0;
       k2/tau2, -1/tau2 ];
G = [ k1/tau1; 0 ];
H = [0 1];
J = 0;

sysC = ss(F,G,H,J);
Gc   = tf(sysC);

% Ts vía polo más rápido
fn = max(abs(zpk(Gc).P{1}))/pi;
Tn = 1/fn;
Ts = Tn/4;

sysD = c2d(sysC, Ts, 'zoh');
[A,B,C,D] = ssdata(sysD);

%% --- Chequeo
n = size(A,1);

%% --- Polos objetivo en z ---
p_ctrl = [0.8 + 1j*0.2; 0.8 - 1j*0.2];    % CONTROL
K = acker(A,B,p_ctrl);

% Prefiltro para el caso sin integrador
[~,~,Nbar] = refi(A,B,C,K);

%% --- Observadores (mismo set rápido) ---
p_obs = [0.2+1j*0.2  0.2-1j*0.2];
L_pred   = acker(A', C',        p_obs).';    % estimador de PREDICCIÓN
L_actual = acker(A',(C*A)',     p_obs).';    % estimador ACTUAL

%% --- Integrador Ogata 6.19 con tus fórmulas ---
m = 1;
Ahat = [A B; zeros(m,n+m)];
Bhat = [zeros(n,m); eye(m)];
Chat = [C zeros(1,m)];

polos_i = [p_ctrl.' 0.6];     % mismos 2 + uno real
Khat = acker(Ahat,Bhat,polos_i);

Aux  = [A-eye(size(A))  B;
        C*A             C*B];
K2K1 = (Khat + [zeros(1,n) eye(m)]) / Aux;
K2   = K2K1(1,1:n);           % sobre x
K1   = K2K1(1,n+1:end);       % sobre integrador

```

```
% observadores para el caso con integrador
p_obs_int = [0.2+1j*0.2 0.2-1j*0.2];
L_pred_i   = acker(A', C',        p_obs_int).';
L_actual_i = acker(A',(C*A)',     p_obs_int).';

%% ================== SIMULACIÓN ==================
N  = 200;
t  = 0:Ts:(N-1)*Ts;
r  = ones(1,N);       % escalón 1
r(1:30) = 0;          % arranca en 0

% ruido en medición
w  = [zeros(1,75),0.01*ones(1,125)];     % ajustá el nivel

% ----- 1) PREDICTIVO SIN integrador -----
X_p   = zeros(n,N);
Xh_p  = zeros(n,N);
U_p   = zeros(1,N);

% ----- 2) ACTUAL SIN integrador -----
X_a   = zeros(n,N);
Xh_a  = zeros(n,N);
U_a   = zeros(1,N);

% ----- 3) PREDICTIVO CON integrador -----
Xi_p   = zeros(n,N);
Xhi_p  = zeros(n,N);
Vi_p   = zeros(1,N);
Ui_p   = zeros(1,N);

% ----- 4) ACTUAL CON integrador -----
Xi_a   = zeros(n,N);
Xhi_a  = zeros(n,N);
Vi_a   = zeros(1,N);
Ui_a   = zeros(1,N);

for k = 1:N-1
    rk = r(k);

    %% ===== 1) PREDICTIVO (sin int) =====
    y_p = C*X_p(:,k);
    u_p = Nbar*rk - K*Xh_p(:,k);
    X_p(:,k+1)  = A*X_p(:,k)  + B*u_p+ w(k);
    Xh_p(:,k+1) = A*Xh_p(:,k) + B*u_p + L_pred*(y_p - C*Xh_p(:,k));
    U_p(k)      = u_p;

    %% ===== 2) ACTUAL (sin int) =====
    u_a = Nbar*rk - K*Xh_a(:,k);
    X_a(:,k+1) = A*X_a(:,k) + B*u_a+ w(k);
    y_a        = C*X_a(:,k+1) ;          % medición en k+1
    z_a        = A*Xh_a(:,k) + B*u_a;
    Xh_a(:,k+1)= z_a + L_actual*(y_a - C*z_a);
    U_a(k)     = u_a;

    %% ===== 3) PREDICTIVO + integrador =====
    yi_p          = C*Xi_p(:,k) ;
    Vi_p(k+1)     = Vi_p(k) + (rk - yi_p);        % integrador real
    ui_p          = K1*Vi_p(k+1) - K2*Xhi_p(:,k);
    Xi_p(:,k+1)   = A*Xi_p(:,k) + B*ui_p+ w(k);
    Xhi_p(:,k+1)  = A*Xhi_p(:,k) + B*ui_p + L_pred_i*(yi_p - C*Xhi_p(:,k));
    Ui_p(k)       = ui_p;

    %% ===== 4) ACTUAL + integrador =====
    yi_a          = C*Xi_a(:,k);
    Vi_a(k+1)     = Vi_a(k) + (rk - yi_a);
    ui_a          = K1*Vi_a(k+1) - K2*Xhi_a(:,k);
```

```
    Xi_a(:,k+1)    = A*Xi_a(:,k) + B*ui_a + w(k);
    yia_next       = C*Xi_a(:,k+1) + w(k);        % misma muestra de ruido para comparar
    z_ai           = A*Xhi_a(:,k) + B*ui_a;
    Xhi_a(:,k+1)   = z_ai + L_actual_i*(yia_next - C*z_ai);
    Ui_a(k)        = ui_a;
end

%% ====== OFFSET AC ======
off = 2.024;
X_p   = X_p   + off;   Xh_p  = Xh_p  + off;   U_p  = U_p  + off;
X_a   = X_a   + off;   Xh_a  = Xh_a  + off;   U_a  = U_a  + off;
Xi_p  = Xi_p  + off;   Xhi_p = Xhi_p + off;   Ui_p = Ui_p + off;
Xi_a  = Xi_a  + off;   Xhi_a = Xhi_a + off;   Ui_a = Ui_a + off;
r_plot = r + off;

%% ================== GRAFICAS ==================

% 1) PREDICTIVO: sin vs con integrador
figure;
subplot(3,1,1);
plot(t, X_p(1,:), 'b', t, Xi_p(1,:), 'r'); grid on;
ylabel('x_1');
legend('pred','pred+int');
title('Estimador predictivo -- sin vs con integrador'); grid minor;

subplot(3,1,2);
plot(t, X_p(2,:), 'b', t, Xi_p(2,:), 'r', t, r_plot, 'k--'); grid on;
ylabel('x_2 / y'); grid minor;

subplot(3,1,3);
plot(t(1:end-1), U_p(1:end-1), 'b', t(1:end-1), Ui_p(1:end-1), 'r'); grid on;
ylabel('u'); xlabel('t'); grid minor;

% 2) ACTUAL: sin vs con integrador
figure;
subplot(3,1,1);
plot(t, X_a(1,:), 'b', t, Xi_a(1,:), 'r'); grid on;
ylabel('x_1'); grid minor;
legend('act','act+int');
title('Estimador actual -- sin vs con integrador');

subplot(3,1,2);
plot(t, X_a(2,:), 'b', t, Xi_a(2,:), 'r', t, r_plot, 'k--'); grid on; grid minor;
ylabel('x_2 / y');

subplot(3,1,3);
plot(t(1:end-1), U_a(1:end-1), 'b', t(1:end-1), Ui_a(1:end-1), 'r'); grid on; grid minor;
ylabel('u'); xlabel('t');

% 3) esfuerzos todos
figure;
plot(t(1:end-1), U_p(1:end-1),  'b', ...
     t(1:end-1), Ui_p(1:end-1), '--b', ...
     t(1:end-1), U_a(1:end-1),  'r', ...
     t(1:end-1), Ui_a(1:end-1), '--r'); grid on;
legend('pred','pred+int','act','act+int','Location','Best');
title('Esfuerzos de control');
xlabel('t'); ylabel('u');  grid minor;

%% ========== MAPA DE POLOS RESUMIDO (3 GRAFICOS GRANDES) ==========
figure('Position',[100 100 1000 900]);  % figura grande

% === 1) Sistema sin integrador ===
subplot(1,3,1);
zgrid; hold on;
%axis equal; axis([-1.2 1.2 -1.2 1.2]);

```

```
p_planta = eig(A);
p_cl     = eig(A - B*K);

plot(real(p_planta), imag(p_planta), 'ko', 'MarkerSize',8, 'LineWidth',1.4);
plot(real(p_cl),     imag(p_cl),     'rx', 'MarkerSize',10, 'LineWidth',1.6);

title('Sistema sin integrador');
%legend({'Planta A','Lazo cerrado A - BK'},'Location','best');
xlabel('Re\{z\}'); ylabel('Im\{z\}');
grid on; box on;

% === 2) Sistema con integrador ===
subplot(1,3,2);
zgrid; hold on;
%axis equal; axis([-1.2 1.2 -1.2 1.2]);

p_aug    = eig(Ahat);
p_aug_cl = eig(Ahat - Bhat*Khat);

plot(real(p_aug),    imag(p_aug),    'm^', 'MarkerSize',8, 'LineWidth',1.4);
plot(real(p_aug_cl), imag(p_aug_cl), 'c*', 'MarkerSize',10, 'LineWidth',1.6);

title('Sistema con integrador (Ahat y Ahat - BhatKhat)');
%legend({'Aumentado Ahat','Aum. cerrado Ahat - BhatKhat'},'Location','best');
xlabel('Re\{z\}'); ylabel('Im\{z\}');
grid on; box on;

% === 3) Estimadores (uno solo, mismo polos para ambos) ===
subplot(1,3,3);
zgrid; hold on;
%axis equal; axis([-1.2 1.2 -1.2 1.2]);

p_obs = eig(A - L_pred*C);    % mismo para predictivo y actual

plot(real(p_obs), imag(p_obs), 'bs', 'MarkerSize',9, 'LineWidth',1.5);

title('Estimadores de estado (p_{obs})');
%legend({'A - LxC'},'Location','best');
xlabel('Re\{z\}'); ylabel('Im\{z\}');
grid on; box on;

sgtitle('Mapa de polos -- Sistemas y estimadores');
```

Listing 1. `HojaDeCalculos.m`: cálculo de parámetros, diseño (K, N_bar, L, K_1, K_2) y generación de gráficos (respuestas y mapas de polos).

B. *PSoC — Estimador predictivo (C)*

```c
#include "project.h"
#include "config.h"

/* ---------- Helpers ---------- */
static inline uint8_t volt_to_dac(float v)
{
    if (v < SAT_MIN) v = SAT_MIN;
    if (v > SAT_MAX) v = SAT_MAX;
    float n = v / SAT_MAX; if (n < 0) n = 0; if (n > 1) n = 1;
    return (uint8_t)(255.0f * n);
}
static inline float sat_u_phys(float u)
{
    if (u > SAT_MAX) u = SAT_MAX;
    if (u < SAT_MIN) u = SAT_MIN;
    return u;
}

/* ---------- Modelo discreto ---------- */
float a11 = 0.932581f, a12 = 0.000000f;
```

```c
float a21 = -1.145618f, a22 = 0.455938f;
float b1  = -0.117824f, b2  = 0.080093f;
float c1  = 0.0f,       c2  = 1.0f;

/* ---------- Ganancias de control CON integrador ----------
 * u[k] = k1 * v[k]  -  (k21 * xhat1 + k22 * xhat2)
 * v[k] = v[k-1] + (r[k] - y[k])
 */
float k21 = -1.0207f;   /* K2 sobre xhat1 */
float k22 = 0.6242f;   /* K2 sobre xhat2 */
float k1  = 0.2280f;   /* K1 sobre el integrador */

/* ---------- Observador (predictivo) ---------- */
float l1  = -0.503374f, l2  = 0.988519f;

/* ---------- Estado ---------- */
static volatile uint8_t tick_flag = 0;
static volatile float ref_ac = 0.0f;
static float xhat1 = 0.0f, xhat2 = 0.0f;
static float vint  = 0.0f;        /* integrador v[k] */

static const float ref_max =  0.5f * A_AMPL;
static const float ref_min = -0.5f * A_AMPL;

/* ---------- ISR de muestreo ---------- */
CY_ISR(adquirirMuestra)
{
    /* referencia AC según el switch */
    ref_ac = ref_state_Read() ? ref_max : ref_min;

    /* DAC auxiliar para scope */
    VDAC8_ref_SetValue(volt_to_dac(ref_ac + VDDA2));

    tick_flag = 1u;
}

/* ---------- MAIN ---------- */
int main(void)
{
    CyGlobalIntEnable;

    /* Front-end analógico */
    Opa_dac_Start();
    Opa_stage1_Start();
    Opa_vdda_2_Start();
    Opa_stage2_Start();

    VDAC8_Start();
    VDAC8_ref_Start();

    VADC_Start();  // Firmware trigger (NO free-run)

    muestra_disponible_StartEx(adquirirMuestra);
    timer_ref_Start(); // genera Ts

    for (;;)
    {
        if (tick_flag)
        {
            tick_flag = 0u;

            /* 1) Medir salida física y[k] (ya en AC) */
            float y_ac = VADC_CountsTo_Volts(VADC_GetResult16());

            /* 2) Integrador: v[k] = v[k-1] + (r[k] - y[k]) */
            /* si querés anti-windup, lo podés saturar acá */
            float err_ry = ref_ac - y_ac;
```

```
            vint += err_ry;



            /* 3) Control con integrador:
             * u_ac = k1 * v  -  (k21 * xhat1 + k22 * xhat2)
             */
            float u_ac = k1 * vint - (k21 * xhat1 + k22 * xhat2);

            /* 4) Observador de PREDICCION:
             * xhat+ = A*xhat + B*u + L*(y - C*xhat)
             */
            float y_hat = c1 * xhat1 + c2 * xhat2;
            float e_y   = y_ac - y_hat;

            float x1n   = a11 * xhat1 + a12 * xhat2 + b1 * u_ac + l1 * e_y;
            float x2n   = a21 * xhat1 + a22 * xhat2 + b2 * u_ac + l2 * e_y;

            xhat1 = x1n;
            xhat2 = x2n;

            /* 5) DAC físico (con saturación y offset) */
            float u_phys = sat_u_phys(u_ac + VDDA2);
            VDAC8_SetValue(volt_to_dac(u_phys));
        }
    }
}
```

C. *PSoC — Estimador actual (C)*

```c
#include "project.h"
#include "config.h"   // SAT_MIN, SAT_MAX, VDDA2, A_AMPL

/* ========= Helpers ========= */
static inline uint8_t volt_to_dac(float v)
{
    if (v < SAT_MIN) v = SAT_MIN;
    if (v > SAT_MAX) v = SAT_MAX;
    float n = v / SAT_MAX; if (n < 0) n = 0; if (n > 1) n = 1;
    return (uint8_t)(255.0f * n);
}
static inline float sat_u_phys(float u_phys)
{
    if (u_phys > SAT_MAX) u_phys = SAT_MAX;
    if (u_phys < SAT_MIN) u_phys = SAT_MIN;
    return u_phys;
}

/* ========= Modelo discreto ========= */
static const float a11 = 0.932581f, a12 = 0.000000f;
static const float a21 = -1.145618f, a22 = 0.455938f;
static const float b1  = -0.117824f, b2  = 0.080093f;
/* C = [0 1] */

/* ========= Ganancias =========
   u[k] = k0 * v[k]  -  (k1 * xhat1 + k2 * xhat2)
   v[k] = v[k-1] + (ref[k] - y[k])
   (esto es tu diseño con integrador)
*/
static const float k1 = -1.0207f;    // sobre xhat1
static const float k2 = 0.6242f;    // sobre xhat2
static const float k0 = 0.228f;    // sobre el integrador

/* ========= Observador (actual) =========
   xhat = z + L*(y - C*z)
```

```
*/
static const float l1 = -0.503374f;
static const float l2 =  0.988519f;

/* ========= Estado global ========= */
static volatile uint8_t tick_flag = 0;
static volatile float   ref_ac    = 0.0f;
static float xhat1 = 0.0f, xhat2 = 0.0f;
static float v_int = 0.0f;                  // <-- integrador

static const float ref_max =  0.5f * A_AMPL;
static const float ref_min = -0.5f * A_AMPL;

/* ========= ISR ========= */
CY_ISR(adquirirMuestra)
{
    ref_ac = ref_state_Read() ? ref_max : ref_min;
    VDAC8_ref_SetValue(volt_to_dac(ref_ac + VDDA2));
    tick_flag = 1u;
}

/* ========= main ========= */
int main(void)
{
    CyGlobalIntEnable;

    /* Analógico */
    Opa_dac_Start();
    Opa_stage1_Start();
    Opa_vdda_2_Start();
    Opa_stage2_Start();

    VDAC8_Start();
    VDAC8_ref_Start();

    VADC_Start();  // firmware trigger

    isr_tick_StartEx(adquirirMuestra);
    timer_ref_Start();

    for(;;)
    {
        if (tick_flag)
        {
            tick_flag = 0u;

            /* ===== (a) Medir y[k] para el integrador ===== */
            VADC_StartConvert();
            while (VADC_IsEndConversion(VADC_RETURN_STATUS) == 0) { }
            float y_k = VADC_CountsTo_Volts(VADC_GetResult16());   // ya AC

            /* ===== (b) Actualizar integrador: v[k] = v[k-1] + (r - y) ===== */
            float err_ry = ref_ac - y_k;
            v_int += err_ry;
            /* opcional anti-windup
            if (v_int > 5.0f)  v_int = 5.0f;
            if (v_int < -5.0f) v_int = -5.0f;
            */

            /* ===== (c) Control con integrador =====
               u_ac = k0 * v_int  -  (k1 * xhat1 + k2 * xhat2)
            */
            float u_ac = k0 * v_int - (k1 * xhat1 + k2 * xhat2);

            /* ===== (d) Enviar al DAC (con offset) ===== */
            float u_phys = sat_u_phys(u_ac + VDDA2);
            VDAC8_SetValue(volt_to_dac(u_phys));
```

```
103
104             /* esfuerzo realmente aplicado en AC */
105             float u_ac_apl = u_phys - VDDA2;
106
107             /* ===== (e) Predicción: z = A*xhat + B*u ===== */
108             float z1 = a11 * xhat1 + a12 * xhat2 + b1 * u_ac_apl;
109             float z2 = a21 * xhat1 + a22 * xhat2 + b2 * u_ac_apl;
110
111             /* ===== (f) Medición y[k+1] para la corrección ACTUAL ===== */
112             VADC_StartConvert();
113             while (VADC_IsEndConversion(VADC_RETURN_STATUS) == 0) { }
114             float y_k1 = VADC_CountsTo_Volts(VADC_GetResult16());
115
116             /* ===== (g) Corrección: xhat = z + L*(y[k+1] - C*z) ===== */
117             float innov = y_k1 - z2;          // C = [0 1]
118             xhat1 = z1 + l1 * innov;
119             xhat2 = z2 + l2 * innov;
120         }
121     }
122 }
```