Trabajo de Aplicación

Tomás Vidal, Lautaro Frangi, Thomas Sille
Grupo 10
Control Automático III
Facultad de Ingeniería, UNLP, La Plata, Argentina.
25 de Noviembre, 2024.

I. Introducción

El siguiente informe describe la realización del trabajo de aplicación asignado a los alumnos de control automatico III, el mismo está dividido en dos secciones la primera que tiene que ver con la identificación de sistemas para una planta dada y una segunda en donde se realizara un paso a paso de la construcción de un un controlador PID para la misma.

II. Identificación de sistema

II-A. Simulación del sistema

En esta instancia, se simuló la planta con el objetivo de poder efectuar un análisis de su comportamiento. Esto se realizó en el programa de diseño de circuitos electrónicos y digitales Proteus Design Suite, el mismo nos permitió diseñar y corroborar el correcto funcionamiento de los algoritmos implementados para la síntesis de las diferentes señales de entrada a la planta. Debido a que la planta cuenta con seis componentes capaces de almacenar energía (capacitores), la misma puede modelarse como un sistema lineal de sexto orden. Cada etapa está separada por un amplificador operacional realimentado negativamente y alimentado con cinco voltios.

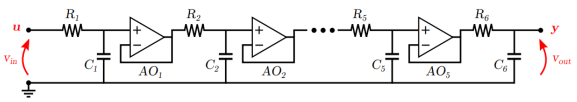

Fig. 1. Esquemático de la placa

II-B. Respuesta al escalón

Se empleó una señal PWM con ciclo de trabajo del 20 %, para hacer que la entrada vaya de 0V a 1V (en el segundo 0), luego de que se estableciera la salida (a los 3 segundos), se cambió el PWM al 80 % de manera que actuará como escalón, y luego de pasados 3 segundos, se volvió a llevar el ciclo de trabajo al 20 %. Esto se realizó tanto en simulación (en Proteus), como en la placa física, como los datos son idénticos para ambos casos, de ahora en adelante se referirá solamente a los datos.

II-C. Modelos de la planta

Una vez hecho el relevamiento de los datos, se emplearon para obtener un modelo de la planta original de alto orden,

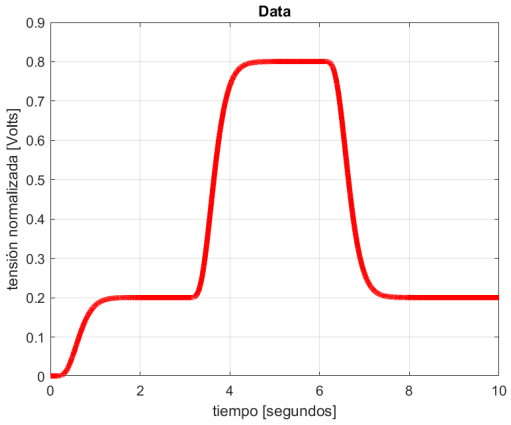

Fig. 2. Respuesta al escalón del sistema (*normalizada a 5V*)

uno del tipo FOPDT[1] y otro SOPDT[2], para esto utilizamos la aplicación de Matlab System Identification[3] habiendo previamente procesados los datos (se tuvo que linealizar el tiempo).

II-C.1. Modelo FOPDT:

$$\frac{e^{-0{,}33791s}}{0{,}28802s+1} \tag{1}$$

II-C.2. Modelo SOPDT:

$$\frac{e^{-0{,}24087}}{(0{,}18561+1)(0{,}18561+1)} \tag{2}$$

II-C.3. Modelo de orden 6:

$$\frac{1{,}261*10^6}{s^6+75{,}11s^5+1869s^4+2{,}523104s^3+1{,}888105s^2+7{,}568105s+1{,}26106} \tag{3}$$

Tipo	EFP	EMC	fit porcentual
FOPDT	0.001008	0.01007	91.97 %
SOPDT	0.0002428	0.0002421	96.06 %
Orden 6	$3{,}963*10^{-9}$	$3{,}912*10^{-9}$	99,98 %

TABLA I. Precisión de los modelos

II-C.4. Precisión de los modelos:

Es importante tener en cuenta que estos modelos están **normalizados** con

[1]First order plus dead time: primer orden más tiempo muerto
[2]Second order plus dead time: segundo orden más tiempo muerto
[3]Herramiento de MATLAB Toolbox que permite hacer identificación de sistemas

respecto a 5V.

Para corroborar la precisión de los modelos, se obtuvieron más datos con otro escalón, con los que se hicieron las verificaciones y cálculos. Los resultados de estos modelos se presentan en la figura 3, para hacer una mejor interpretación de los datos y los modelos, se centró la respuesta al escalón de los datos entre 1V y 4V, y entre 3 y 6 segundos.

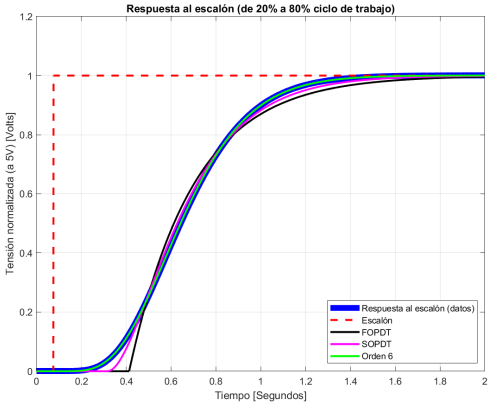

Fig. 3. Respuesta al escalón de los modelos

II-C.5. Análisis de robustez: Considerando una tolerancia de los elementos del circuito de un $10\,\%$ (al ser el producto de estos, termina siendo la suma, es decir una tolerancia del $20\,\%$) y partiendo del modelo de orden 6, es posible establecer una familia de modelos a partir de un peso de incertidumbre dinámica global. Se consideró un peso de **incertidumbre aditiva**, ya que el mismo es el más adecuado a la hora de aproximar un modelo de orden elevado con otro de orden reducido. La condición a evaluar para el diseño del peso es la siguiente:

$$|G(jw) - G_0(jw)| < |W\delta(jw)|$$

Donde G es la familia de plantas y G_0 se obtiene a partir del modelo FOPDT obtenido. Entonces graficando esta condición para distintos valores para la constante de tiempo de G se obtuvo la siguiente función peso, compuesta de un cero y un polo:

$$G_0 = \frac{e^{-0{,}33791s}}{0{,}28802s + 1}$$

$$W_a = \frac{0{,}25s}{0{,}45s + 1}$$

De manera tal que nuestra familia de plantas queda de la siguiente forma

$$G(s) = g_0 + (1 + W_a\delta) \quad , \quad |\delta| \leq 1$$

$$G(s) = \frac{e^{-0{,}33791s}}{0{,}28802s + 1} + (1 + \frac{0{,}25s}{0{,}45s + 1)}\delta) \quad , \quad |\delta| \leq 1$$

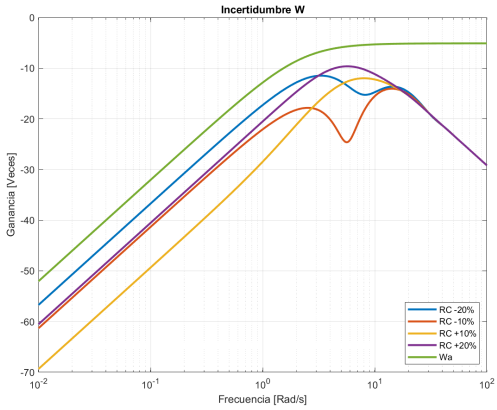

Fig. 4. Incertidumbre y modelo W_a

II-D. *Señal pseudoaleatoria*

Se empleó una señal pseudoaleatoria como entrada al sistema para hacer la identificación del mismo. Se tomó como periodo de la señal $T_{PRBS} = 200ms$ y una cantidad de bits $N = 7$, con esto se tiene un tiempo de ensayo $D_s = 25{,}4$. Estos valores cumplen con la frecuencia de *Nyquist* y el tiempo de *rising* del sistema $t_r \cong 1{,}4segundos$ (ancho de banda $BW = \frac{1}{t_r} \cong 0{,}71Hz$). *Nuevamente las tensiones se normalizaron con respecto a 5V.*

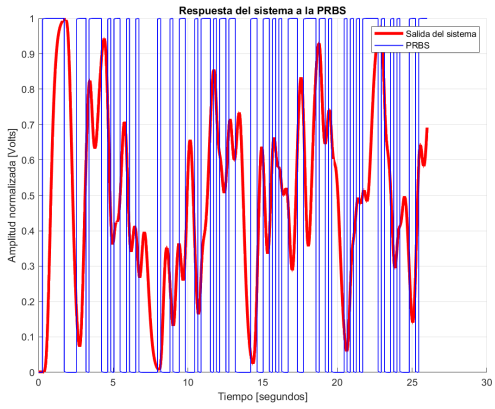

Fig. 5. Respuesta del sistema a la PRBS

Luego se empleó la herramienta *System Identification* para obtener los mismos modelos que antes. Los modelos obtenidos se muestran a continuación y, como se puede observar, son mucho peor que los primeros.

II-D.1. Modelo FOPDT:

$$\frac{e^{-0{,}2948s}}{0{,}3772s + 1} \tag{4}$$

II-D.2. Modelo SOPDT:

$$\frac{e^{-0{,}20498}}{(0{,}20383 + 1)(0{,}20383 + 1)} \tag{5}$$

II-D.3. Modelo de orden 6:

$$\frac{1{,}261 * 10^6}{s^6 + 2{,}652s^5 + 109s^4 + 209{,}1s^3 + 3777s^2 + 4121s + 4{,}084 * 10^4} \tag{6}$$

Tipo	EFP	EMC	fit porcentual
FOPDT	0.01969	0.01962	42,74 %
SOPDT	0.01656	0.01648	47,53 %
Orden 6	0.3103	0.0001459	50,2 %

TABLA II. Precisión de los modelos

III. Diseño de controlador PID

En la siguiente sección se presenta el diseño de un controlador del tipo PID de la planta mostrada en la primer sección, haciendo énfasis en la teoría de controladores PID en sistemas con Retardo. Dicho Controlador fue implementado en un microcontrolador ATMega328p sobre el cual se efectuaron ensayos de forma experimental de forma física y virtual a través de la herramienta de software Proteus 8 Professional. Para el diseño del mismo se utilizó el modelo de primer orden obtenido en la primer sección:

$$G_1(s) = \frac{e^{(-0{,}33791s)}}{0{,}28802s + 1}$$

III-A. PID con predictor Smith

Se parte del diseño de un controlador PI aplicado a la planta definida como:

$$P_n(s) = \frac{K_p e^{-Ls}}{Ts + 1}$$

$$G_n(s) = \frac{K_p}{Ts + 1}$$

$$C(s) = \frac{K_1(T_1 s + 1)}{T_1}$$

Al sintonizar el sistema para cancelar el polo del modelo de lazo abierto, se obtiene el controlador equivalente:

$$C_e(s) = \frac{K_1(Ts + 1)}{Ts + K_1 K_p(1 - e^{-Ls})}$$

Usando una aproximación de Padé de primer orden para el retardo y equiparando a un controlador PID en formato serie con filtrado en la acción derivativa, se obtiene:

$$C_e(s) = \frac{K_c(T_I s + 1)(T_D s + 1)}{T_I s(1 + \alpha T_D s)}$$

$$\alpha = \frac{1}{1 + \frac{L}{T_0}}$$

$$K_c = \frac{T}{(L + T_0)K_p}$$

Siendo el valor de K_p es la ganancia de la planta, L es el retardo y T_I es la constante de tiempo, por lo tanto solo se tendrá que sintonizar el valor de T_0.
Para la sintonización de T_0, se diseña el sistema en Simulink[4]. Se evalúa la respuesta al escalón considerando una perturbación también en escalón, aplicada a los 5 segundos.

[4]Herramienta provista en el Toolbox de MATLAB.

Primero probamos con un valor pequeño de T0=0.1, donde se observa un sobrepeso significativo en la respuesta (línea amarilla), aunque la perturbación se rechaza. Sin embargo, la acción de control (línea verde) resulta excesiva, dificultando su implementación en un Arduino.

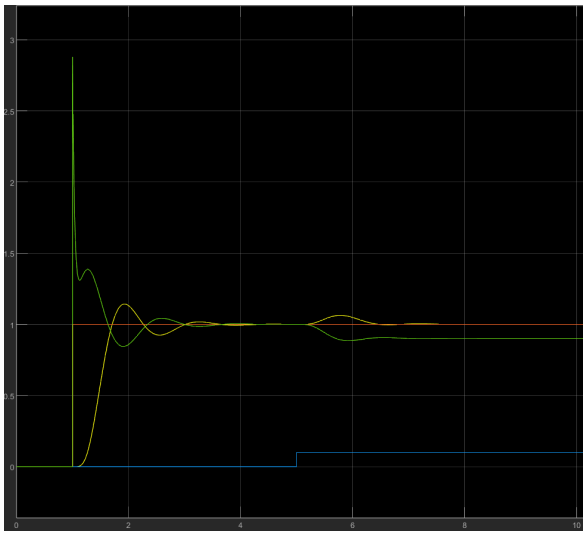

Fig. 6. Simulación en Simulink $T_0 = 0{,}1$

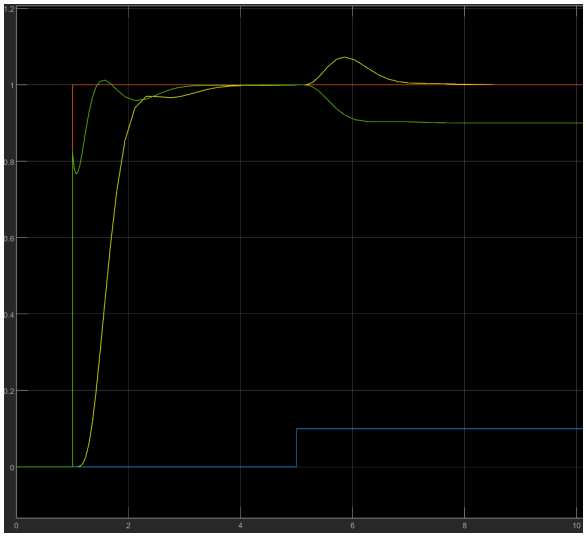

Fig. 7. Simulación en Simulink $T_0 = 0{,}35$

Aumentando el valor de T_0 a 0.35, se puede observar que con este ajuste, el sobrepaso desaparece, pero la magnitud de la acción de control sigue siendo alta.Aunque hay que aclarar que el rechazo a la perturbación, debido a la cancelación del polo dominante del sistema, que aparece en la transferencia entre la perturbación y la salida, no se podrá rechazar más rápido que la constante de tiempo de la planta.

III-B. Filtro

Para reducir la acción de control y mejorar la velocidad de respuesta sin sobrepaso, se implementa un filtro en la entrada

de la referencia. El filtro usado es el siguiente:

$$F(s) = \frac{(1 + \beta T_r s)}{(1 + T_r s)}$$

El filtro se conecta en serie con la referencia, como se muestra en el diagrama correspondiente:

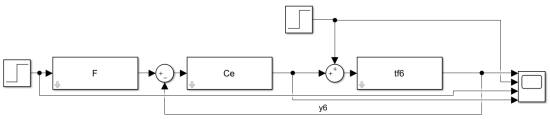

Fig. 8. Diagrama en bloques del sistema en Simulink

Al simular nuevamente con $T_0 = 0{,}35$, observamos que los picos de la acción de control disminuyen significativamente, lo que indica una mejora en la respuesta. Sin embargo, para garantizar un rango de seguridad adicional y evitar riesgos de inestabilidad o sobrecarga del controlador, se decidió incrementar ligeramente el valor de $T_0 = 0{,}35$.

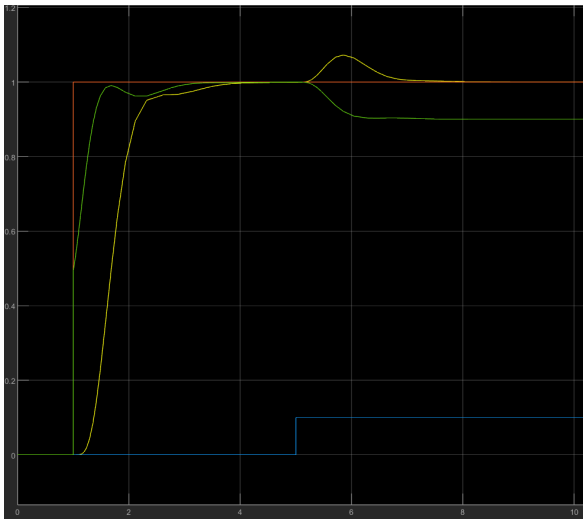

Fig. 9. Simulación en Simulink $T_0 = 0{,}35$

Con $T_0 = 0{,}4$, la acción de control no presenta picos significativos, lo que permite implementarla en un Arduino sin problemas. Además, el rechazo a la perturbación es suficientemente rápido, con el pico ocurriendo entre 5.2 y 5.7 segundos, es decir, un tiempo de 0.5 segundos.

Dado que la constante de tiempo de la planta es T=0.28 segundos, este tiempo de rechazo es adecuado, ya que no puede ser inferior debido a la cancelación del polo dominante del sistema. Reducir aún más T_0 podría hacer la acción de control demasiado agresiva, generando sobrepasos u oscilaciones. Por lo tanto, $T_0 = 0{,}4$ es un valor equilibrado para cumplir con los requisitos del diseño.

III-C. Comprobación de robustez

El controlador diseñado demostró cumplir de forma satisfactoria con las especificaciones de diseño para el modelo estimativo de la planta. Sin embargo, es necesario establecer

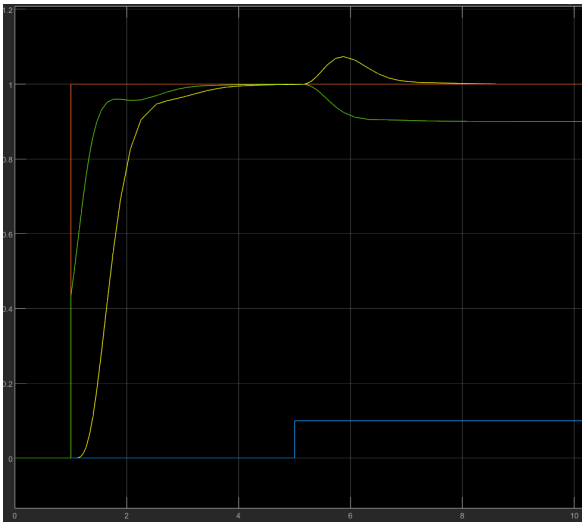

Fig. 10. Simulación en Simulink $T_0 = 0{,}4$

un peso de incertidumbre adecuado para representar el error de modelado presente. Considerando el peso de incertidumbre aditiva propuesto en la sección anterior, se obtiene que la condición necesaria para asegurar robustez es la siguiente:

$$|C(jw)W(jw)| < |1 + C(jw)G_0(jw)|$$

En la siguiente gráfica se puede evaluar esta condición para el controlador diseñado.

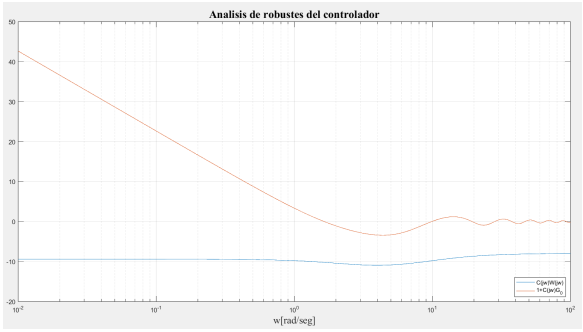

Fig. 11. Comprobación de condición de robustez

En la gráfica de validación, se observa que esta condición se cumple, garantizando robustez en el diseño.

III-D. Discretización y simulación

Por último, se desarrolló un código en Arduino para implementar el controlador discreto. Para ello, se emplearon aproximaciones numéricas tanto para la derivada como para la integral. Una vez sintonizado el modelo PID en formato serie, se convirtió el controlador al formato paralelo antes de realizar la discretización. Esto se realizó porque trabajar con las ecuaciones en formato paralelo resulta más conveniente y práctico para este propósito.

$$C_e(s) = \frac{(K_p + \frac{K_I}{s} + K_D s)}{(1 + sT_f)}$$

Donde:

$$T_f = 0{,}5 * \alpha * L$$

$$K_I = \frac{K_c}{T_I}$$

$$K_D = 0{,}5 * L * K_c$$

$$K_p = Kc\frac{(T + 0{,}5 * L)}{T}$$

Partiendo de la relación entre la salida de la acción de control U(s) y el error E(s):

Al aplicar la transformada inversa de Laplace y despejar, se obtiene la siguiente ecuación diferencial, que será la base para la discretización:

$$u(t) + T_f u'(t) = K_p e(t) + K_I \int e(t)dt + K_D e'(t)$$

A partir de esta expresión, se discretizan los términos de la siguiente manera: Término proporcional (P):

$$P = K_p e(t) \rightarrow P[k] = K_p e[k]$$

Término derivado asociado a la salida (D1):

$$D1 = T_f u'(t) \rightarrow D1[k] = T_f \frac{(u[k] - u[k-1])}{h}$$

Término integral (I):

$$I = K_I \int e(t)dt \rightarrow I' = K_I e \rightarrow I[k+1] = h{*}K_I e[k]{+}I[k]$$

Término derivado asociado al error (D2):

$$D2 = K_D e'(t) \rightarrow D2[k] = K_D \frac{(e[k] - e[k-1])}{h}$$

Sumando todos los términos, se obtiene la ecuación en diferencias que describe el controlador discreto:

$$u[k] = \frac{(e[k](K_p + \frac{K_D}{h}) - e[k-1]\frac{K_D}{h} + I[k] + \frac{T_f}{h} * u[k-1])}{1 + \frac{T_f}{h}}$$

Esta ecuación fue implementada en el código de Arduino. Antes de programar el controlador en el microcontrolador, se realizó una simulación en Proteus para verificar su correcto funcionamiento. A continuación, se muestra una imagen de la simulación realizada:

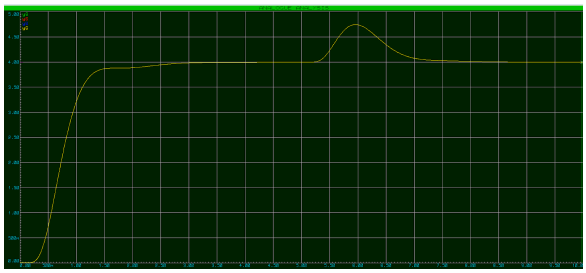

Fig. 12. Simulación en Proteus

III-E. Ensayo experimental

Finalmente el programa fue cargado en un Arduino UNO, obteniéndose resultados experimentales consistentes con las simulaciones en MATLAB y Proteus, validando la correcta implementación y sintonización.

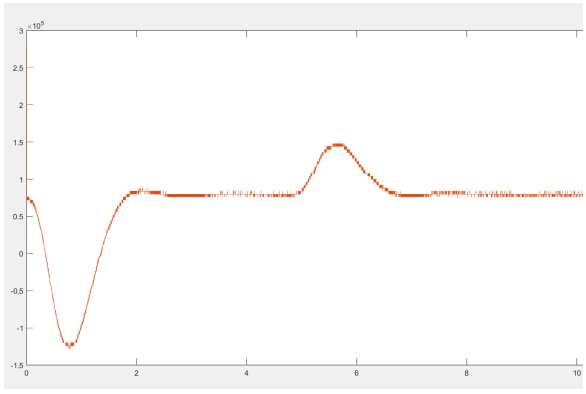

Fig. 13. Resultado experimental

IV. Conclusión

HACER CONCLUSIÓN!