PRÁCTICA N°3: CONTROL II. CONTROL PID POR UBICACIÓN DE POLOS.

(Noviembre de 2023)

Dago Mauricio Quiroz Hoyos
dagom.quirozh@uqvirtual.edu.co
Universidad Del Quindio
Armenia, Quindio

Resumen-
En esta práctica se llevó a cabo el diseño e implementación del control PID por ubicación de polos en un sistema emulado de segundo orden, se partió por obtener la respuesta del sistema en lazo abierto, luego se diseñó e implementó el control PID en donde se corrigio el error estático de posición y se buscó mejorar el te y el Mp, por último la obtención de los datos se realizó con ayuda de labVIEW .

I. INTRODUCCIÓN

El laboratorio número 3 de sistema de control automático 2 busca la implementación de un control PID por ubicación de polos, siendo este fundamental para el diseño y la mejora de los sistemas dinámicos. Esta sesión, se realiza desde la respuesta en lazo abierto del sistema hasta la implementación y semejanza del control PID tanto en emulación como en simulación.

Esto se lleva a cabo mediante el uso de herramientas educativas y de código abierto, tanto a nivel de software como de hardware, por ejemplo Matlab, labVIEW, el entorno de desarrollo integrado (IDE) de Arduino y el microcontrolador ESP 32; con el objetivo de encontrar la representación de la función de transferencia en proporcional, integral y derivativo.

II. MÉTODOS E INSTRUMENTOS

A. INSTRUMENTOS.

Matlab es un entorno de programación y software de alto rendimiento utilizado en ingeniería, matemáticas y ciencias para análisis numérico, modelado y visualización de datos.[1]

LabVIEW es un entorno de desarrollo de software y hardware creado por National Instruments, diseñado para la programación de sistemas de medición y control. Se destaca por su interfaz gráfica de programación, que utiliza un lenguaje llamado G, facilitando la creación de aplicaciones de adquisición de datos, automatización y pruebas.[2]

IDE(Arduino).
2 Microcontroladores (Esp 32).

B. MÉTODOS.

Para esta práctica de laboratorio se asignó una función de transferencia de segundo orden:

$$G(s) = \frac{0.143230S+0.052084}{S^2+0.242424S+0.057871} \quad (1)$$

El siguiente paso es discretizar Gs; con el comando cd2 en matlab se lleva al dominio z:

```
Gz =

  0.01441 z - 0.01389
  -------------------
  z^2 - 1.975 z + 0.976

Sample time: 0.1 seconds
```

Figura 1. Representación de la función de transferencia en el dominio Z.

En la figura 1 se observa que el tiempo de muestreo es de 0.1 segundo, este se debe tener en cuenta para la implementación en simulación como en emulación.

Se obtienen los valores del numerador y del denominador para hallar la ecuaciones de diferencia y poder implementar el resultado en variables de estado.

$b0 = 0.014407733553361$
$b1 = -0.013893180866316$
$a1 = -1.975477362671597\ 7$
$a2 = 0.97604908678172$
$Gz = \frac{b0Z + b1}{Z^2\ a1Z\ \ a2}$
(2)

Al igual que en [3] se lleva esta representación de ecuaciones de diferencia a variables de estado.

$$Y(Z) = b0u\left(z^{-1}\right) + b1\,u\left(z^{-2}\right) - a1y\left(z^{-1}\right) - ay\left(z^{-2}\right) \qquad (3)$$

Para verificar la correcta discretización del sistema, se lleva a cabo una configuración de bloques en Simulink, como se muestra en la figura 2. Este montaje incluye un bloque step, que genera un escalón de 0 a 1. A continuación, se emplea un bloque transfer Fcn, donde se especifican el numerador y el denominador de la ecuación (1) del sistema. Además, se incorpora otro bloque con Discrete transfer Fcn, en el que se introduce la ecuación (2), indicando también el tiempo de muestreo, establecido en 100 ms. Finalmente, las salidas de estos bloques se conectan a un scope, donde ambas representaciones deben coincidir.

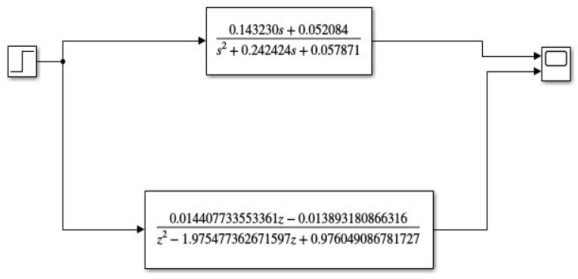

Figura 2. Representación de los sistemas en simulink.

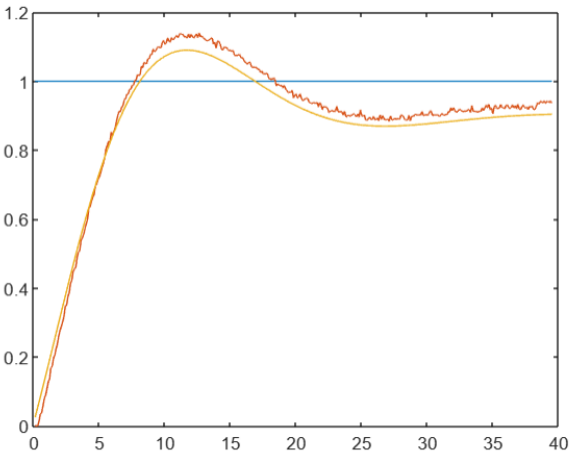

Figura 3. Emulación del sistema en lazo abierto y representación del sistema en simulink.

Para la implementación del control por ubicación de polos se realizó un script en matlab con los datos obtenidos en lazo abierto, se establece tiempo de establecimiento, el porcentaje de sobreimpulso y la ganancia del sistema.

```
>> te

te =

  33.000033000033000

>> Mp

Mp =

   0.160001576336119

>> k

k =

   0.900001727981200
```

Figura 4. G,Mp y te del sistema en lazo abierto.

Se diseña para un te de 30 segundos y un porcentaje de establecimiento del 0.15%; de ahí se parte parte para encontrar el valor de sigma Wd y W, ya con estas variables encontradas se procede determinar los polos, esto se realiza con el comando:

Pds=tf([1],[1 2*sigma_n Wn_n_cuadrado])

Luego se multiplica por el polo flotante y de esa manera obtenemos P1,P2 y P3.

P1 =

1.266666666666667

P2 =

0.333195903779551

P3 =

0.066529237112884

El siguiente paso es hallar PID, es decir el valor constante que acompaña al bloque proporcional, integral y derivativo, para esto se realiza la división en matlab de la siguiente forma.

Kp=(P1-Gs_den(2))/Gs_num(3);
Ki=(P2-Gs_den(3))/Gs_num(3);
Kd=P3/Gs Núm(3);

Se procede a implementar el control en los bloques de simulink, para conocer la respuesta del sistema con este nuevo conjunto de bloques, como se observa en la figura 5.

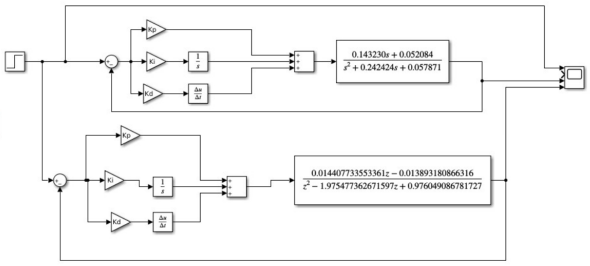

Figura 5. PID por ubicación por polos Simulink.

En la figura 6 se observa la respuesta obtenida del control PID por ubicación de polos.

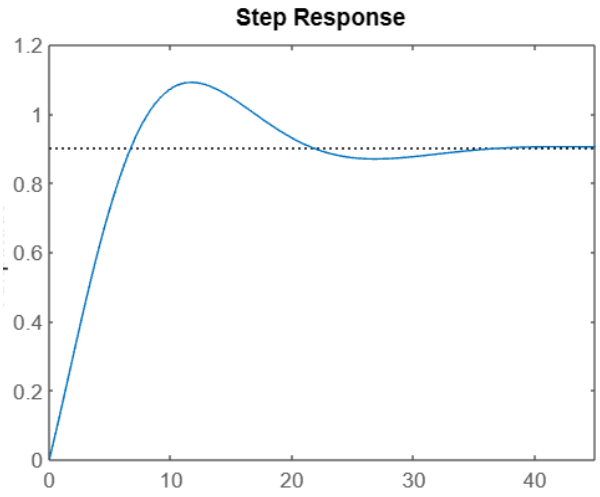

Figura 6. Respuesta del sistema controlado.

Para la toma de datos y la implementación de de la simulación del sistema de segundo orden, se siguen los pasos realizados en los anteriores laboratorios, en la figura 7 se encuentra implementado en el sistema listo para ser emulado.

Además se puede observar cómo se realiza la toma de los datos obtenidos, la inclusión en de un reloj y de la parte simulada como emulada figura 7.

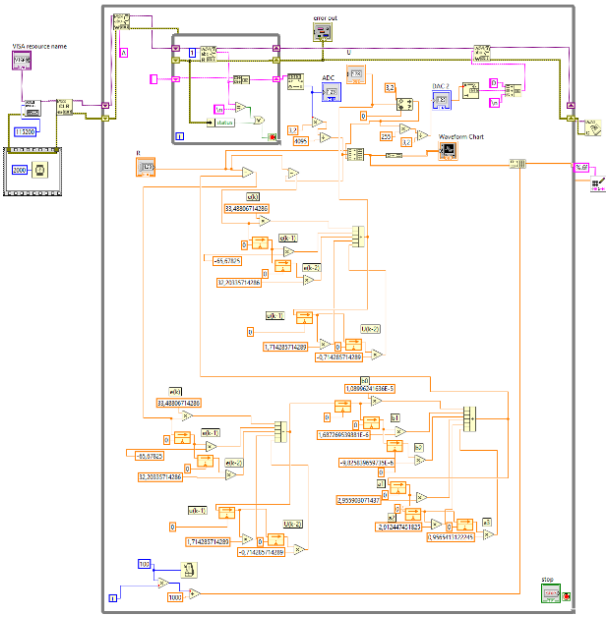

Figura 7. Implementación del controlador PID para el sistema.

Para esta simulación son necesarias dos Esp 32 con una como tarjeta de adquisición de datos y la otra como planta. El código es el mismo utilizado en los laboratorios anteriores, solo se realizan los cambios de bn y an, en la figura 7 se observa la respuesta obtenida por labVIEW en matlab.

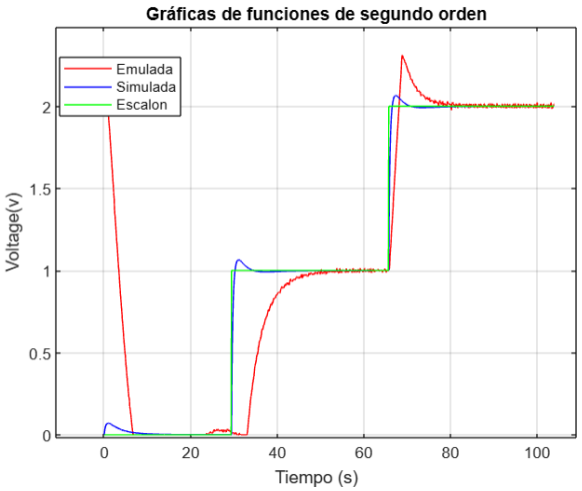

Figura 8. Respuesta del sistema emulado con el sistema simulado.

III. RESULTADOS Y DISCUSIÓN

Para analizar ambas respuestas de una forma más ágil y práctica, se propone una tabla comparativa (tabla 1) que demuestra los valores más relevantes de la figura 7, como lo son el tiempo de establecimiento, la ganancia y el sobreimpulso.

Tabla 1. Tabla comparativa entre ambas respuestas.

	G	te (s)	Mp (%)
Sistema simulado	1	6	10
sistema emulado	1	13	0

Mediante esta tabla, es posible concluir que el control implementado cumple con el objetivo de eliminar el error estático de posición, pero además, mejora el sobreimpulso aunque sacrifica un poco el tiempo de establecimiento. De igual manera, como se muestra en la figura 7 ante un primer escalon la respuesta corresponde a un control adecuado, al igual que en el segundo escalón, aunque presenta un comportamiento distinto en cuanto al sobre impulso y con esto al tiempo de establecimiento.

IV. CONCLUSIONES

1. El manejo de herramientas computacionales permiten una mayor exactitud en cálculos, pero idealmente la matemática, se ajusta a los que establecemos, lo importante es saber con las limitaciones que cuenta mi entorno, que para el caso del sistema es el voltaje que es de 0 a 3.2 voltios aproximadamente.
2. Las respuestas obtenidas del control son más precisas ante escalones bajitos, ya que es particularmente efectivo para cambios pequeños porque combina una acción proporcional inmediata, una acumulación de errores a lo largo del tiempo y una anticipación de la tendencia del error. Estos elementos permiten una respuesta rápida y precisa del sistema ante perturbaciones o cambios pequeños en la entrada.

V. BIBLIOGRAFÍA

- [1] MathWorks.(s.f). Matemáticas. Gráficas, Programación. MATLAB - El lenguaje del cálculo técnico (mathworks.com)
- [2] Product. (s. f.). https://www.ni.com/es-co/shop/product/labview.html.
- [3] Control2/Laboratorio #1_ EMULACION DE UN SISTEMA DE TERCER ORDEN (1).pdf at main · Dago02/Control2 (github.com)

VI. ANEXOS

Script

```
clear all
clc
format long
%% función de transferencia dada
Gs =tf([0.143230 0.052084],[1 0.242424 0.057871]); %
%% Discretizando el sistema con Tm=0.1
Gz=c2d(Gs,0.1);
%% calculando te, Mp y k
Gs_num=cell2mat(Gs.num);
Gs_den=cell2mat(Gs.den);
sigma=Gs_den(2)/2;
te=4/sigma;
Wd=sqrt(Gs_den(3)-sigma^2);
Mp=exp(-pi*sigma/Wd);
k=Gs_num(3)/Gs_den(3);
Mp_n=0.15;
te_n=30;

sigma_n=4/te_n;
Wd_n=-pi*sigma_n/log(Mp_n);
Wn_n_cuadrado = sigma_n^2+Wd_n^2;
Pds=tf([1],[1 2*sigma_n Wn_n_cuadrado]);
Raices_Pds=roots(cell2mat(Pds.num));
Pds1=tf([1],[1 1]);
Pd=Pds*Pds1;
Pd_den=cell2mat(Pd.den);
P1=Pd_den(2);
P2=Pd_den(3);
P3=Pd_den(4);

%% Hallando los valores de Kp, Ki y Kd para el PID
Kp=(P1-Gs_den(2))/Gs_num(3);
Ki=(P2-Gs_den(3))/Gs_num(3);
Kd=P3/Gs_num(3);
```

```
%% Hallando los valores para la discreticación del PID
%Para la parte integral
b=0.1*Ki;
%para la parte derivativa
b1=(4*Kd)/1.4;
b2=(-4*Kd)/1.4;
a=1/1.4;
%% Gráfica de la función de transferencia
figure (1)
step(Gs)
%% Toma de datos del sistema emulado
datos1=xlsread("Dago_datos.csv");

Tiempo=datos(:,1);
Ref=datos(:,2);
Y=datos(:,3);
U=datos(:,4);
```