PRÁCTICA N°5: CONTROL II.
CONTROL RST.
(Noviembre de 2023)

Dago Mauricio Quiroz Hoyos, Mariana Jimenez Duarte, Laura Valentina Buitrago
dagom.quirozh@uqvirtual.edu.co, mariana.jimenezd@uqvirtual.edu.co, laurav.buitrago@uqvirtual.edu.co
Universidad Del Quindio
Armenia, Quindio

Resumen-
En este laboratorio se espera implementar un control RST en un sistema de prueba (hélice con un grado de libertad) mediante técnicas operadas en diferentes softwares que permiten visualizar en primera instancia una respuesta simulada y posteriormente la respuesta del sistema real, logrando además, visualizar gráficamente como el control responde ante diferentes escalones aplicados en tiempo real.

I. INTRODUCCIÓN

En esta práctica se requiere añadir un control RST en un sistema de prueba de elección propia, en este caso hacia una hélice con un grado de libertad, en la cual se debe poner a prueba su comportamiento una vez sea incorporado el control, esperando obtener una respuesta ya conocida mediante un simulación previa al momento de la verificación. Lo que se espera es que una vez el sistema alcance su punto de equilibrio, definido con anterioridad al momento de analizar la planta, pueda seguir una referencia que es cambiada a discreción.

Con esta implementación se busca obtener la capacidad de a partir de un sistema propio, encontrar los lineamientos del control y posteriormente implementarlo al sistema, lo cual se cumple a cabalidad al poder contemplar como el brazo móvil, que como su nombre lo especifica, solo tiene un grado de libertad, sigue la referencia y se estabiliza en dicho valor en un tiempo y con unas características muy similares con las cuales fue diseñado el control.

Para hacer posible dicho control, fue necesario utilizar herramientas de simulación gráficas como MatLab y LabVIEW, la primera de ellas para modelar el sistema real en uno lineal y obtener una respuesta a esperar, y además, para comprobar si el sistema real coincide con el simulado, la segunda permite conectar el sistema real con un entorno de simulación para variar la referencia y ver como esta se comporta. Todo este procedimiento nos permite entender con mayor claridad el comportamiento son control de la planta y evidenciar de manera visual tanto en la planta como en gráficas, como mejora su respuesta.

A continuación, se aprecia la estructura del documento:

1. Introducción.
2. Métodos e instrumentos.
3. Resultados y discusión.
4. Conclusiones.
5. Referencias.
6. Anexos

II. MÉTODOS E INSTRUMENTOS

A. INSTRUMENTOS.

- MATLAB(Simulink).

Plataforma que combina programación con cálculos matemáticos para analizar una variedad de datos y crear modelos que se ajusten a esos datos. Contiene diferentes herramientas en diferentes demostraciones, lo que lo hace adecuado para control y muchas otras áreas. [1]

- LabVIEW.

Entorno de programación gráfico que permite realizar pruebas y mediciones automatizadas. Tiene una amplia gama de herramientas que lo hacen útil para muchas aplicaciones de ingeniería. [2]

- IDE(Arduino).
- Microcontrolador (Arduino uno).
- Hélice con un grado de libertad.

B. MÉTODOS.

La implementación de este control se realiza en un sistema de prueba (hélice con un grado de libertad). Primero se necesita encontrar una función de

transferencia que represente el sistema, el proceso se realiza mediante herramientas de software, por un lado se utiliza Matlab con la APP system identification, y por otro lado se obtienen los datos necesarios para ingresar a la herramienta mediante un script en el IDE de Arduino. Los valores de pwm que pasan a través del puente H son diferentes, es decir, se utilizan distintos escalones en los puntos de equilibrio Y=168 y U=133, por lo que el sistema logra una buena adquisición de datos.

La variable controlada es el voltaje suministrado al motor por el microcontrolador, y lo que se espera controlar es la lectura del potenciómetro. De [3] se deben seguir los procedimientos más detallados para obtener las respuestas de la Figura 1.

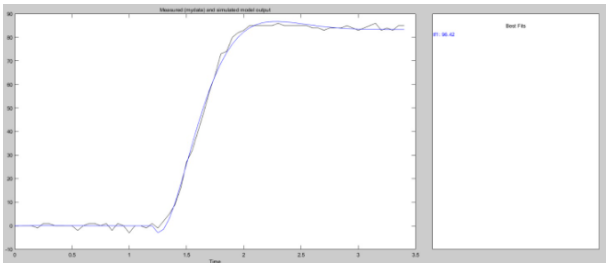

Figura 1. Representación de la función de transferencia de la planta.

Se parte de la función de transferencia hallada para el sistema real (hélice con un grado de libertad):

Gs = tf([-1.778627076821461 25.737756427243420],[1 6.338208167305483 19.439754070643087]) (1)

Se cambia de dominio Gs a discreto con el comando c2d, en él se trabaja con un tiempo de muestreo de 0.1 segundos hallando (2).

Gz = tf([-0.023840353499535 0.211419112101930],[1 -1.388882317419561 0.530560747954682]) (2)

Se diseña el control con base en los siguiente datos originales:

Mp=5 % te=1.5 $\sigma = 2.66$

El polo deseado para este sistema hallado anteriormente en [4] expresado en el dominio z es :

Zd = 0.62398311987±0.24489595799i (3)

Se desea eliminar el error estático de posición, para ello se debe agregar una integral que a su vez representa un polo más a la derecha del polo deseado pero dentro del círculo unitario, por estas razones el polo añadido es 0.5.

A partir de esto se obtiene el polinomio deseado característico mediante las siguientes líneas de código:

```
pd=(z-r1z)*(z-r2z)*(z-0.5)*(z-0.5)
Pd=pd.num{1}
```

Donde la variable num {1} es el numerador de la función de transferencia del sistema en el dominio z.

Pd= z^4 - 2.248 z^3 + 1.947 z^2 - 0.7613 z + 0.1123(4)

Para obtener los valores de los bloques T, S, y R, en la figura 2 se observa la topología de control RST con integral, para hallar los valores de s1, t0, r0, r1y r2; se implementó un código en matlab, en donde se halla la matriz ecnit y igint, por último se multiplican estas matrices con ecint inversa y se obtiene un vector columna con los valores de s1,r0,r1 y r2, y para finalizar se halla t0 con la suma de r0, r1 y r2.

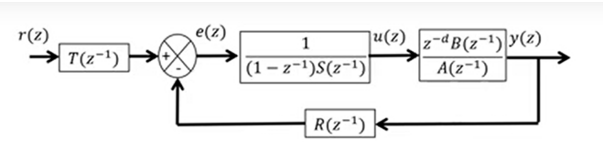

Figura 2. Esquema de control RST con integral[5].

En la figura 3 se observa la implementación del control RST en bloques de simulink, para representar la integral se implementa el bloque discrete filter, al igual que para ingresar los valores de S y R, y se establece en cada uno de estos bloques el tiempo de muestreo inicial de 0.1 para obtener la respuesta esperada.

s1 = 0.187312337719481 r2 = 1.001389197633898
r0 = 1.946122990199160 t0 = 0.268370904417488
r1 = -2.679141283415570

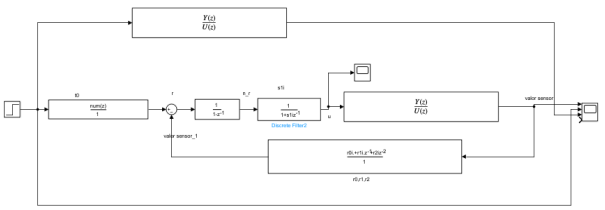

Figura 3. Implementación del sistema en simulink.

En la figura 4 se observan dos respuestas una en lazo abierto (naranja) y la otra lazo cerrado (amarillo) con el control RST.

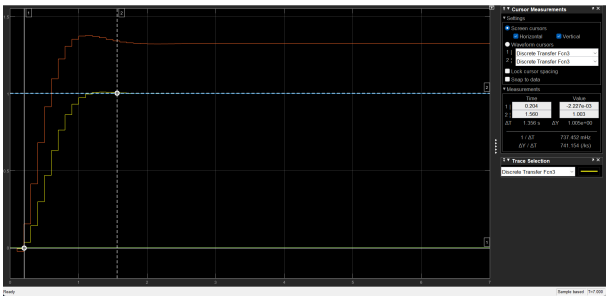

Figura 4. Respuesta simulada del control RST.

Con la ayuda de los cursores se puede observar el tiempo de establecimiento el cual es de 1.356 segundos, y se evidencia que se eliminó el error estático de posición.

Para implementar este control al sistema real se modifica la función timers que es donde ocurre la realimentación y el proceso bloque a bloque.

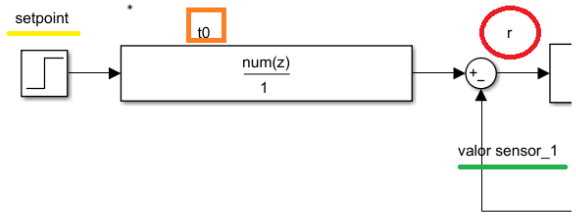

Figura 5. Sesión uno de la implementación.

$$\frac{r(Z)}{setpoint(z)} = t0$$
$$r = t0 * setpoint(z)$$
(5)

En la figura 5 se observa que el valor setpoint se debe multiplicar por la constante t0 esto va al sumador y se resta con el valor obtenido en la realimentación de la salida Y, para encontrar el valor de salida que en este caso es r.

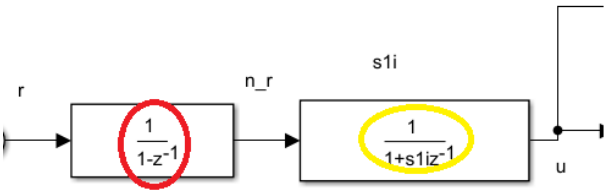

Figura 6. Sesión dos de la implementación.

En la primera parte encerrada en rojo de la figura 6 se puede observar un retraso, el cual se va a ingresar a r y por esta razón a la salida tenemos n_r. Del bloque encerrado en amarrilo tenemos que:

$$\frac{n\_r(Z^{-1})}{U(Z)} = \frac{1}{1+S1(Z^{-1})}$$

$$U(Z) = n\_r(Z^{-1}) - U(Z^{-1})$$
(6)

En este punto el siguiente bloque es el que se va a ingresar al sistema, por esta razón se pasa del modelo real al lineal, por lo cual se debe sumar el punto de equilibrio de la U, que para esta implementación es de 133, además a la salida de este bloque se obtiene la Y del sistema que para la planta es el valor del sensor (potenciometro).

En la figura 7 se observa el bloque de realimentación, en el cual se tiene como entrada el valor del sensor y a la salida el valor del sensor con un retraso.

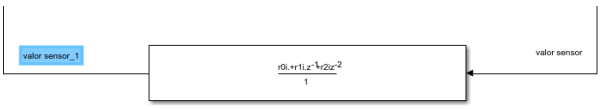

Figura 7. Sesión tres de realimentación.

$$\frac{valorsensor\_1(Z^{-1})}{valorsensor(Z)} = r0 + r1 + r2$$
$$valorsensor\_1 = valorsensor * r0 + valorsensor\_1$$
$$=* r1 + valorsensor\_2 * r2$$
(7)

La respuesta que se espera con esta implementación es la misma que en en la figura 4. Se verificó la coincidencia entre ambas respuestas para poder pasar a la siguiente y última etapa que es la implementación del control a nuestro sistema real.

El código implementado en este laboratorio es similar al que se utilizó en el control mediante compensadores, solo se modifica la función del temporizador en donde se están realizando los pasos que debe sufrir el setpoint y valorSensor respectivamente en la entrada y salida del sistema real para que su respuesta sea la esperada.

El código LabVIEW introducido es el utilizado en [4] para adquirir datos del sistema controlado. La Figura 8 muestra la interfaz del programa. Se cambian los valores del sensor y del sistema variando el pwm. Cuando se alcanza el valor de referencia, se muestra en la pantalla la señal del sistema ya controlado.

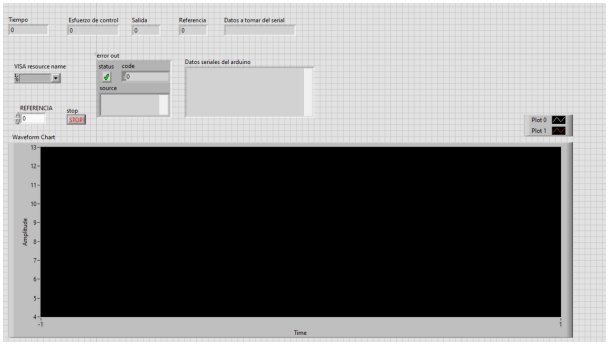

Figura 8 **.** Programa de toma de datos del sistema real .

Luego de obtener los datos del sistema se procede a hacer la comparación entre la respuesta simulada y la real.

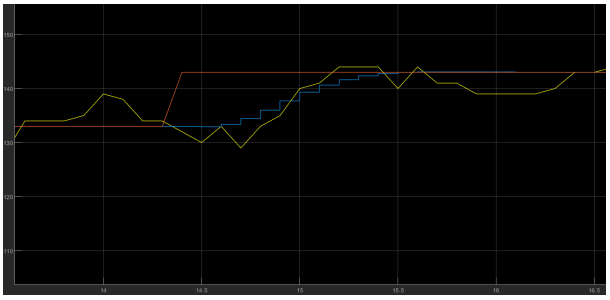

Figura 9. Sistema simulado vs real.

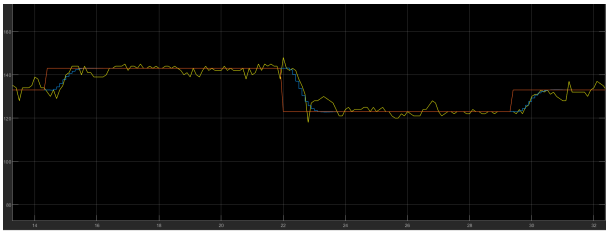

Figura 10. Sistema simulado vs real en vista amplia.

III. ANÁLISIS Y RESULTADOS

Para llevar a cabo un análisis con más detalle se almacenan en las tablas 1 y 2 los parámetros más relevantes de las respuestas mostradas en las figuras 4 y 9, lo cual permite comparar los resultados y concluir si la estrategia de control funciona correctamente en este sistema.

Tabla 1. Comparación entre respuesta en lazo abierto y simulada.

	Simulada	Lazo abierto
MP (%)	1	5.3
te (s)	1.3	1.7
G	1	1

La respuesta en lazo abierto mantiene la ganancia unitaria, por lo cual se conserva en la señal controlada, ahora bien, se evidencia una notable mejoría en el sobre impulso y tiempo de establecimiento, ya que la planta por sí sola maneja unos muy buenos parámetros, que pudieron disminuirse mediante el control.

Es de resaltar que la respuesta en lazo abierto fue analizada sobre la referencia que ella por sí sola toma, ya que no puede seguir la referencia impuesta.

Tabla 2. Comparación entre respuesta simulada y real.

	Simulada	Real
MP (%)	0	1
te (s)	1.3	1.9
G	1	1

Como se observa en la tabla 2, la respuesta del sistema real coincide en gran medida con la señal simulada, manteniendo la forma esperada, además de un tiempo de establecimiento y sobreimpulso similar como se muestra en la figura 10, por lo que se puede concluir que esta estrategia de control funcionó de manera correcta, cumpliendo con la eliminación del error de estado estacionario y manteniendo bajos los parámetros más relevantes del sistema.

IV. CONCLUSIONES

1. Al momento de emular el comportamiento de un sistema es fundamental tener en cuenta los puntos de equilibrio con los cuales fue analizado, esto garantiza una correcta emulación y uso eficiente del sistema para obtener un control esperado.
2. Esta estrategia de control es más poderosa en comparación a otras, ya que permite elegir un polo deseado más rápido que los del sistema aun cuando se tiene un cero en el lado inestable del lugar de las raíces del sistema.
3. Tener un manejo adecuado de los bloques que conforman el sistema controlado permite encontrar de forma más ágil la ecuación que lo rige, la cual es clave para su emulación.
4. Con el control RST se eliminó el error de estado estacionario y se mejoran el sobre

impulso y tiempo de establecimiento, manteniendo la forma inicial de simulación.

5. Al incorporar la integral al diseño del control, fue posible mejorar la respuesta del sistema y además garantizar una estabilidad en la respuesta una vez llega al punto de equilibrio.

V. REFERENCIAS

- [1] MathWorks.(s.f). Matemáticas. Gráficas, Programación. MATLAB - El lenguaje del cálculo técnico (mathworks.com)
- [2] ni.(s.f). ¿Qué es LabVIEW? ¿Qué es LabVIEW? Programación gráfica para pruebas y medidas - NI
- [3] Control2/Laboratorio #1_ EMULACION DE UN SISTEMA DE TERCER ORDEN (1).pdf. Tomado de: at main · Dago02/Control2 (github.com)
- [4] Control2 /Lab4_Control2.docx.pdf. Tomado de: Control2/Lab4_Control2.docx.pdf en la principal · Dago02/Control2 · GitHub (en inglés)
- [5] (281) Control RST con integral - YouTube

VI. ANEXOS

Script matlab:

```
t=0.1;
z= tf('z');
Gs       =       tf([-1.778627076821461
25.737756427243420],[1
6.338208167305483 19.439754070643087])
Gz=c2d(Gs,t)
num=Gz.num{1}
den=Gz.den{1}
%Polos del sistema
r=roots(den)
polo_1=r(1);
polo_2=r(2);
%Ceros del sistema
b=roots(num)
% Obtención del polo deseado
Mp = 0.1;
te =1.3;
sigma = -log(0.02) / te;
wd = -pi * (sigma / log(Mp));
polod= -sigma+wd*1i; %polo deseado
z1=exp(polod*t)    %Determina los polos
en discreta
r1z=0.62398311987+0.24489595799i
r2z=0.623983119870-24489595799i
%Polo deseado
p_d=(z-r1z)*(z-r2z)
P_d=p_d.num{1} %Polo deseado
%Polinomio caracteristico deseado
pd=(z-r1z)*(z-r2z)*(z-0.5)*(z-0.5)
Pd=pd.num{1}
%Matriz
ecint=[1  num(2)  0  0;(den(2)-1)  num(3)
num(2)   0;(-den(2)+den(3))   0   num(3)
num(2);-den(3)  0  0  num(3)]


igint=[(Pd(2)-(den(2)-1));(Pd(3)-den(3
)+den(2));(Pd(4)+den(3));(Pd(5))]
%Se saca la inverza de la matriz para
obtener la columna resultante
res2=inv(ecint)*igint
%s1, r0, r1, r2
s1i=res2(1)
r0i=res2(2)
r1i=res2(3)
r2i=res2(4)
%t0
to2=((r0i)+(r1i)+(r2i))
```