Motor DC y Control

Leslie P. Brenes, Diego Villafuerte

Resumen—En esta práctica se implementaron controladores P (Amplificador), I (Integrador), D (Diferenciador)a un motor. Primero mediante una simulación con Simulink de Matlab y posteriormente con OpAmps.

1. INTRODUCCIÓN

El controlador PID nos permite controlar un sistema en lazo cerrado (donde la salida depende de iteraciones correctivas mediante la retroalimentación) para obtener la salida desada. Se componte de tres elementos: P - Proporcional, I - Integral, D - Derivativa.

Nos interesa aprender a implementar este tipo de controlador para poder realizar correcciones en la señal generada para la alimentación de un motor.

2. MARCO TEÓRICO

La acción de control Proporcional nos permite modificar proporcionalmente la señal de error $e(t)$ esto mediante una constante K_p

Por otro lado, la de control Integral en términos generales acumula iterativamente los errores generados sin embargo tiende a generar inesabilidad en el sistema.

Finalmente, la acción de control Derivativa se determina mediante la constante K_d en relación con la señal de error respecto al tiempo: $K_d \frac{de}{dt}$. De manera general, el control Derivativo lo que permite es corregir el error proporcionalmente.

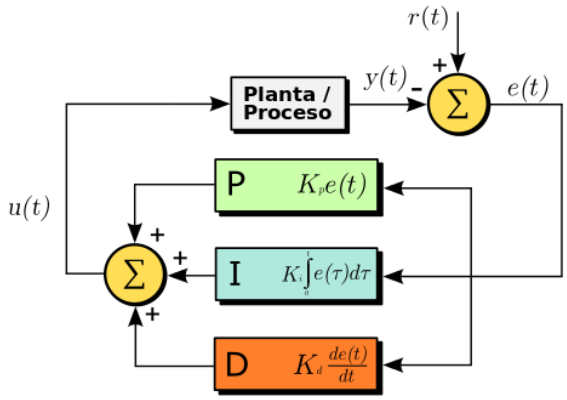

Figura 1. Funcionamiento de un controlador PID.

Los pasos sugeridos para la implementación manual de un control PID son:

1. Aumentar poco a poco la acción proporcional para disminuir el error. (Control P)
2. Aumentar poco a poco la constante derivativa K_d para conseguir estabilidad en la respuesta.
3. Aumentar la constante integral K_i hasta que el error se minimice con la rapidez deseada.

[1]

En particular, existe una heurística para configurar manualmente el controlador PID: el método de Ziegler–Nichols[**?**]. Primero se deben establecer las consantes del Integrador y Derivador en 0. Luego, se incrementa la constante K_p de P hasta llegar a un punto estable. Hacemos $K_p = K_u$ y medimos P_u (el periodo de la oscilación), posteriormente generamos los valores de K_i y K_d. Finalmente editamos las otras constantes de acuerdo a la siguiente tabla 1.

Control	K_p	K_i	K_d
Control P	$\frac{K_u}{2}$		
Control PI	$\frac{K_u^2}{2,2}$	$\frac{P_u}{1,2}$	5415
Control PID	$\frac{K_u}{1,7}$	$\frac{P_u}{2}$	$\frac{K_u}{8}$

Cuadro 1
Tabla de sintonización de Ziegler-Nichols

Los OpAmps son dispositivos de 3 componentes principales: input inversor, input no inversor y output. Cuando se usa de forma lineal, genera la amplificación y/o inversión de una señal. Pero si la configuración es de lazo cerrado puede generar un Integrador o un diferenciador. [2]

3. DESARROLLO

Primero utilizamos el archivo base en Matlab, dentro de Simulink.

El primer paso fue agregar y conectar (virtualmente) los componentes de Amplificación, Integración y Derivación, además tuvimos que modificar uno de los cables para generar la señal de error retroalimentada.

Luego, utilizamos el componente de funciones para Matlab, y para poder usarlo tuvimos que agregar componentes de memorias para guardar los valores que se necesitaban recursivamente: acumulado y fcanterior.

Una vez generada la señal deseada, procedimos a la segunda parte la práctica, que fue mediante OpAmps y un motor.

4. RESULTADOS

En la sección de Simulink, generamos la siguiente configuración (2).

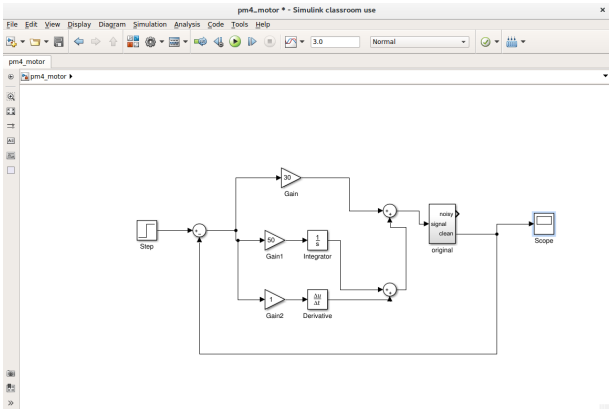

Figura 2. Configuración de componentes PID en Simulink.

Posteriormente, generamos la función correspondiente en Matlab (no pudimos incluir la función generada ya que no contamos con una versión actualizada de Matlab y se generó una incompatibilidad). La configuración fue la siguiente (3).

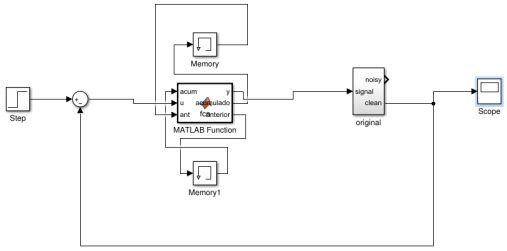

Figura 3. Configuración de componentes PID en Simulink con función de Matlab.

Finalmente, la señal generada fue la siguiente (4).

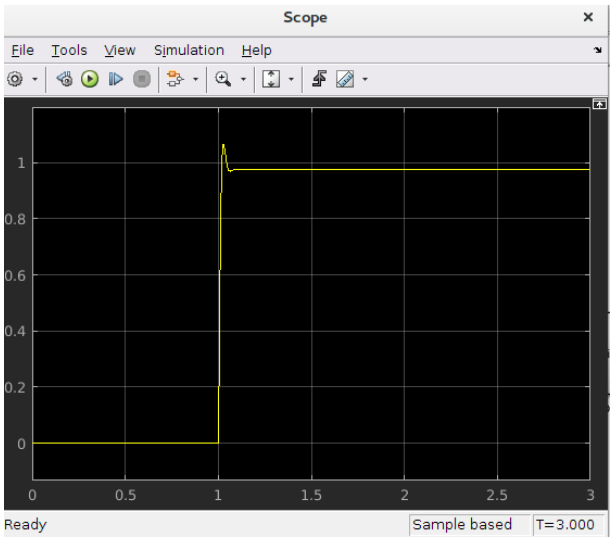

Figura 4. Señal con correcciones mediante PDI.

Para la sección de OpAmps, no solo pudimos generar el amplificador, y no podemos demostrarlo porque antes de tomar fotos se quemó. Después intentamos cambiar de OpAmp pero ya no había TTL081, solo TTL084 con 3 OpAmps integrados, pero no funcionó.

5. CONCLUSIONES

- Leslie: En esta práctica aprendimos a simular un controlador PID mediante Simulink en Matlab, primero aprendimos a agregar los componentes y luego a modificar los parámetros. Luego, vimos como generar la función en lenguaje de Matlab, y tuvimos que agregar las memorias para poder hacer la función recursiva. Finalmente, aunque solo fue a nivel teórico, pudimos entender como funcionan los OpAmps para corregir una señal mediante un control PID, y no solo como componentes aislados.
- Diego: En esta práctica vimos el funcionamiento de OpAmps para la corrección de señales, además aprendimos a usar Simulink y MatLab.

6. ROL O PAPEL

- Leslie: Investigación, configuración de amplificador con OpAmps. Marco teórico.
- Diego: Configuración de PID en Simulink y configuración de OpAmps.

7. FUENTES CONSULTADAS

REFERENCIAS

[1] "Controlador pid." [Online]. Available: https://www.picuino.com/es/arduprog/control-pid.html

[2] "Opamps." [Online]. Available: https://www.electronics-tutorials.ws/opamp/opamp_1.html