Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

TE3001B: Fundamentación de robótica (Gpo 101)

Profesor: Dr. Rigoberto Cerino Jiménez

Actividad 2.1. Sintonización de un controlador PID

Equipo 5

Tania Sofia Arrazola Bello	A01738124
Iñaki Ahedo Madrid	A01738231
Marcos Allen Martínez Cortés	A01737939

24 de febrero de 2026

Resumen

En esta actividad se modeló y simuló en Simulink el motor DC proporcionado por Manchester Robotics, el cual está representado por un sistema dinámico de primer orden. Se realizaron experimentos de respuesta al escalón unitario y a un escalón de mayor amplitud con el objetivo de analizar el comportamiento dinámico del sistema en lazo abierto, identificando su orden, estabilidad, tipo y parámetros característicos como valor final, tiempo de subida y tiempo de establecimiento.
A partir del análisis teórico y experimental se comprobó que el motor presenta un comportamiento estable de primer orden con constante de tiempo $\tau = 0.5$ s y ganancia $K = 1.75$. Posteriormente se realizó la sintonización de un controlador PID para mejorar el seguimiento de referencia y eliminar el error en estado estacionario. Los resultados muestran una mejora significativa en rapidez y precisión respecto al sistema sin control.

Objetivos

El objetivo principal es modelar, simular y analizar el motor DC proporcionado por Manchester Robotics en Simulink, así como sintonizar un controlador PID a partir del análisis dinámico obtenido. Por otro lado los objetivos específicos son:

- Modelar el motor DC utilizando su modelo continuo equivalente.
- Analizar la respuesta al escalón unitario y a un escalón mayor.
- Determinar orden, estabilidad y tipo del sistema.
- Calcular y verificar experimentalmente el valor final y tiempo de establecimiento.
- Diseñar y sintonizar un controlador PID.
- Evaluar la mejora del sistema en lazo cerrado.

Introducción

El funcionamiento del sistema del motor dado por Manchester Robotics era a lazo abierto, debido a que el generador de señales publicaba continuamente una señal senoidal que representa el voltaje o esfuerzo aplicado al motor $u(t)$. Por su parte, el nodo del motor recibía ese impulso y calculaba su velocidad de salida $y(t)$ que estaba únicamente definida por su ganancia K y su constante de tiempo τ. Sin embargo, al carecer de un lazo de retroalimentación que verifique el error existente de la relación entre la entrada y la salida, el motor reaccionaba a la señal senoidal y presentaba un retraso debido a su naturaleza. (Vilanova & Alfaro, 2011)

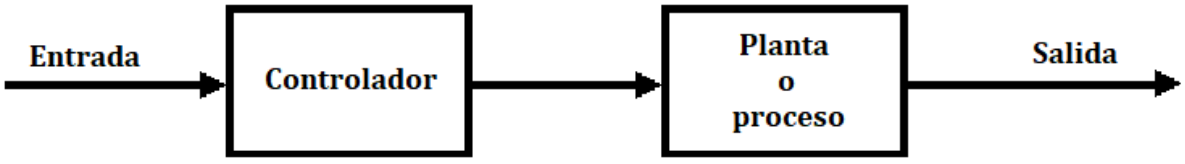

Figura 1: Sistema en lazo abierto.
https://www.researchgate.net/publication/334771046/figure/fig2/AS:786392066039808@1564501887099/Figura-5-Sistema-de-control-en-lazo-abierto.ppm

Para resolver esto y hacer que la señal de salida se pueda ajustar a que sea lo más parecido a la de entrada que en un contexto aplicado, se tiene que aplicar un controlador que tenga una retroalimentación y mida con una diferencia el resultado final con lo que se espera para después ajustar los parámetros y tenga un comportamiento lo más cercano a lo deseado. Para aquello existen varios controladores, en esta actividad se desarrolló un controlador PID. (Vilanova & Alfaro, 2011)

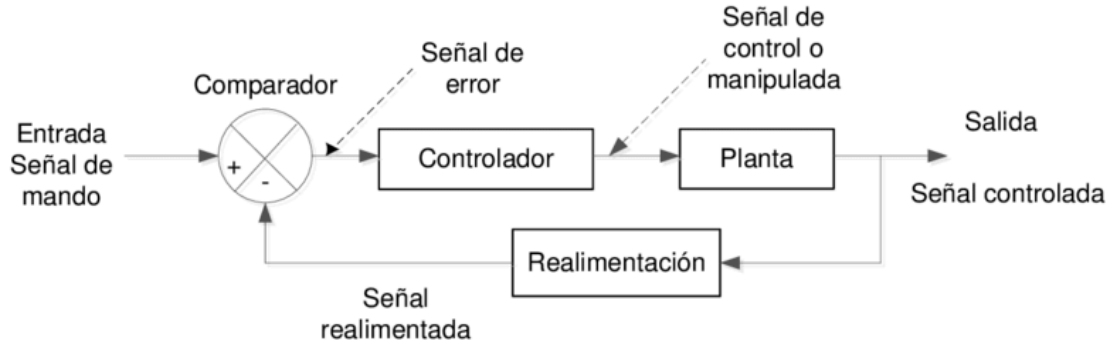

Figura 2: Sistema en lazo cerrado.
https://www.researchgate.net/figure/Diagrama-a-bloques-de-un-sistema-de-control-en-lazo-cerrado_fig1_267454753

Un controlador PID o controlador proporcional, integral y derivativo es un mecanismo de control que funciona mediante retroalimentación. La retroalimentación viene de medir la señal que se produjo a la salida $y(t)$ y compararla con la señal de entrada o el set point $r(t)$ y hacer una diferencia. A esta diferencia se le conoce como error $e(t)$ como se muestra en la ecuación 2:

$$e(t) = r(t) - y(t) \qquad (2)$$

El control PID hace tres acciones con respecto a ese error:

Acción proporcional (K_p): Esta acción es la base del control. Su función es reaccionar a cuánto falta para llegar al valor deseado. Matemáticamente es muy simple debido a que multiplica al error actual por una constante K_p. En el caso de la simulación de un motor, esta constante funciona cuando la velocidad del motor es lejano a la velocidad deseada, la acción de esta constante es enviar un voltaje más alto. Entre menor sea la diferencia entre lo obtenido y lo esperado, la acción de K_p disminuye.

El problema con este controlador es que llega un punto en el que el error es tan pequeño que la señal resultante no tiene la suficiente fuerza para llegar al objetivo, siempre quedándose un poco abajo, esto se le conoce como error de estado estacionario.

Acción integral (K_i): La parte integral del controlador es la que soluciona el problema al solamente usar el control P. Esta acción se encarga de verificar los errores que se han acumulado anteriormente. Mira cuánto tiempo ha pasado en el estado estacionario y va sumando ese error acumulado. Al sumar este error, hace que no se quedé abajo del valor deseado. El problema es que si la ganancia de K_i es muy, el sistema puede llegar a tener oscilaciones hasta volverse inestable.

Acción Derivativa (K_i): La acción derivativa funciona como un amortiguador a los cambios repentinos que tiene el sistema o son causados por la acción proporcional. Este se basa en la razón de cambio del error, es decir, mide la pendiente de esta señal y si esta es muy inclinada, este hace un cambio más intenso y si es más estable, su acción es prácticamente 0. Mientras que las demás acciones se enfocan en eliminar el error, esta acción se enfoca en la estabilidad. Actúa como una fuerza de oposición que frena la velocidad de respuesta del sistema cuando se está aproximando muy rápido al objetivo, evitando que el sistema se vuelva inestable y eliminando las oscilaciones que puede provocar ya sea la señal de entrada o las demás acciones. (Vilanova & Alfaro, 2011)

Si se utilizan las tres señales, se estará utilizando un controlador PID el cual suma estas tres acciones, repercutiendo en la señal de salida como se muestra en la ecuación 3:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \qquad (3)$$

Para sintonizar las constantes de cada acción del controlador PID es importante realizarlo con la mejor precisión posible. Se utilizan varios métodos para la sintonización de estas variables, en esta actividad se consideraron dos métodos que a continuación se presentan:

- **Ziegler-Nichols**

Este método comienza en verificar el momento exacto donde la planta deja de ser estable y empieza a oscilar de forma constante. Para aquello, de igual forma Ki y Kd se inicializan en 0 y se aumenta gradualmente Kp hasta llegar a un punto en donde empezará a oscilar. Debe aumentar este valor hasta encontrar la ganancia crítica K_u, este sería el valor de Kp cuando se logra una oscilación constante. (Åström & Hägglund, 2004)

Después se mide el periodo crítico P_u el cual es el tiempo en segundos que tarda la oscilación en completar un ciclo completo. Una vez encontrados dichos valores, se diseña una tabla de constantes para calcular las tres ganancias del controlador. Estas ganancias son diferentes dependiendo el tipo de controlador que se quiera implementar. A continuación se presenta la tabla de parámetros:

Control	Kp	Ki	Kd
P	$0.5K_u$	-	-
PI	$0.45K_u$	$\frac{1}{12}P_u$	-
PID	$0.6K_u$	$0.5P_u$	$0.125P_u$

Tabla 1: Parámetros de control.

- **PID Tuner App de MATLAB y Simulink**

La aplicación integrada en el software de MathWorks permite ajustar automáticamente (*autotune*) las ganancias para un controlador PID, esto para una planta SISO (Single Input Single Output o, de una entrada y una salida); como resultado se obtiene un tipo de controlador que puede ser especificado o asignado de acuerdo con las ganancias obtenidas (MathWorks, s. f.-a).

La herramienta de MathWorks busca cumplir los objetivos de la sintonización PID: estabilidad del sistema en lazo cerrado, rendimiento adecuado y robustez adecuada

(MathWorks, s. f.-b). El algoritmo utilizado por MathWorks para PID Tuner selecciona por defecto una frecuencia de cruce basada en la dinámica de la planta y un margen de fase objetivo de 60°; y, al modificar parámetros como tiempo de respuesta, ancho de banda o margen de fase, se recalculan las ganancias (MathWorks, s. f.-b). Para un margen de fase mínimo dados, el algoritmo equilibra seguimiento de referencia y rechazo de perturbaciones, permitiendo priorizar uno mediante la opción *DesignFocus*; además, la efectividad de este ajuste depende del número de parámetros del controlador (p. ej. PID en comparación a P o PI) y de las características propias de la planta (MathWorks, s. f.-b).

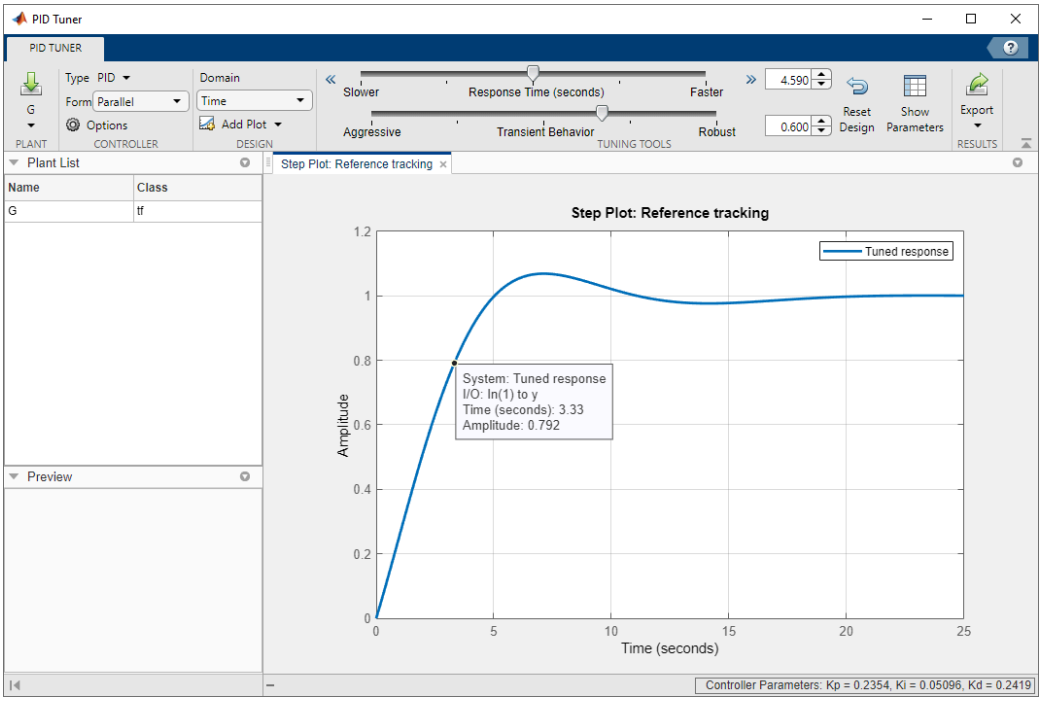

Figura 3: Interfaz de PID Tuner App.

Solución del problema

Motor a simular (nodo ROS)

El motor proporcionado por Manchester Robotics se modela en tiempo discreto como:

$$y[K+1] = y[k] + \left(\frac{-1}{T}y[k] + \frac{K}{T}u[k]\right)T_S$$

El modelo continuo equivalente es:

$$G(s) = \frac{K}{\tau s + 1}$$

Por lo tanto, para esta actividad se utilizó:

$$G(s) = \frac{1.75}{0.5s + 1}$$

Este modelo corresponde a un sistema lineal invariante en el tiempo de primer orden, cuya dinámica está determinada por:

K = 1.75, que es la ganancia estática
τ = 0.5 s que es la constante de tiempo

En lazo abierto, el motor no cuenta con retroalimentación, por lo que su salida depende únicamente de la dinámica interna y de la señal de entrada. Esto provoca retraso y

posible error en estado estacionario cuando se requiere seguir una referencia específica. Para mejorar su desempeño, se buscará implementar un controlador PID, cuya ley de control es:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

donde:

$$e(t) = r(t) - y(t)$$

Es el término proporcional actúa sobre el error actual, el integral elimina el error acumulado y el derivativo mejora la estabilidad amortiguando cambios rápidos.

Por otro lado en el modelo de simulink, la función de transferencia ingresada fue:

$$G(s) = \frac{1.75}{0.5s + 1}$$

Modelado en Simulink (Lazo Abierto)
Se implementó el modelo continuo equivalente utilizando:

- Bloque Step
- Bloque Transfer Function
- Bloque Scope

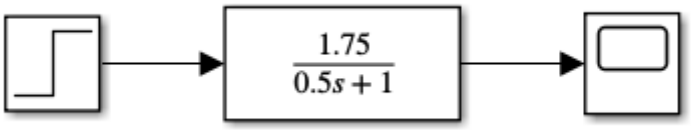

Figura 4: Diagrama en Simulink del sistema en lazo abierto.

Con el modelo implementado, se realizaron dos experimentos variando únicamente la amplitud del escalón.

Experimento 1. Respuesta al escalón unitario
Se midieron los siguientes parámetros:

- **Valor final Teórico**
  Para un sistema de primer orden:
  $$y_{ss} = K \cdot u$$
  que es igual a:
  $$y_{ss} = 1.75 \cdot 1 = 1.75$$

- **Valor en Simulación**
  Obteniendo el valor en simulación: **1.75,** lo cual coincide con el análisis teórico

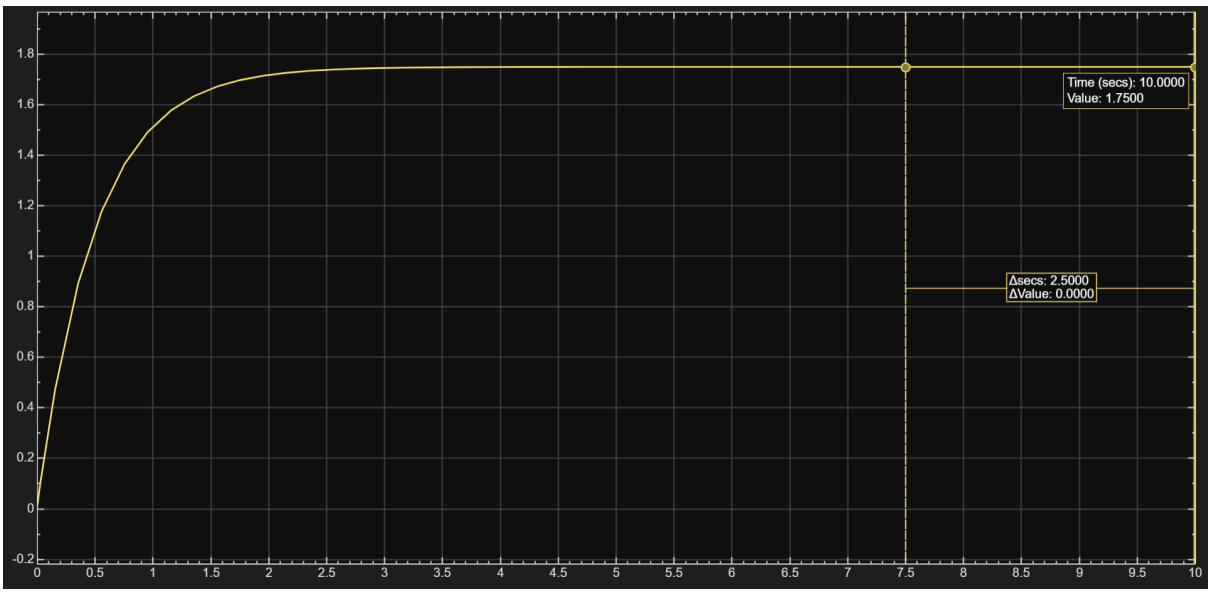

Figura 5: Gráfica obtenida en Simulink mostrando el valor final con entrada de escalón unitario.

- **Tiempo de establecimiento teórico**
  Para sistemas de primer orden:

$$t_s \approx 4\tau$$

$$t_s \approx 4(0.5) = 2s$$

- **Tiempo de Establecimiento simulado**
  El tiempo de establecimiento obtenido fue: **1.9569 s** lo cual indica que el resultado es consistente con la aproximación teórica.

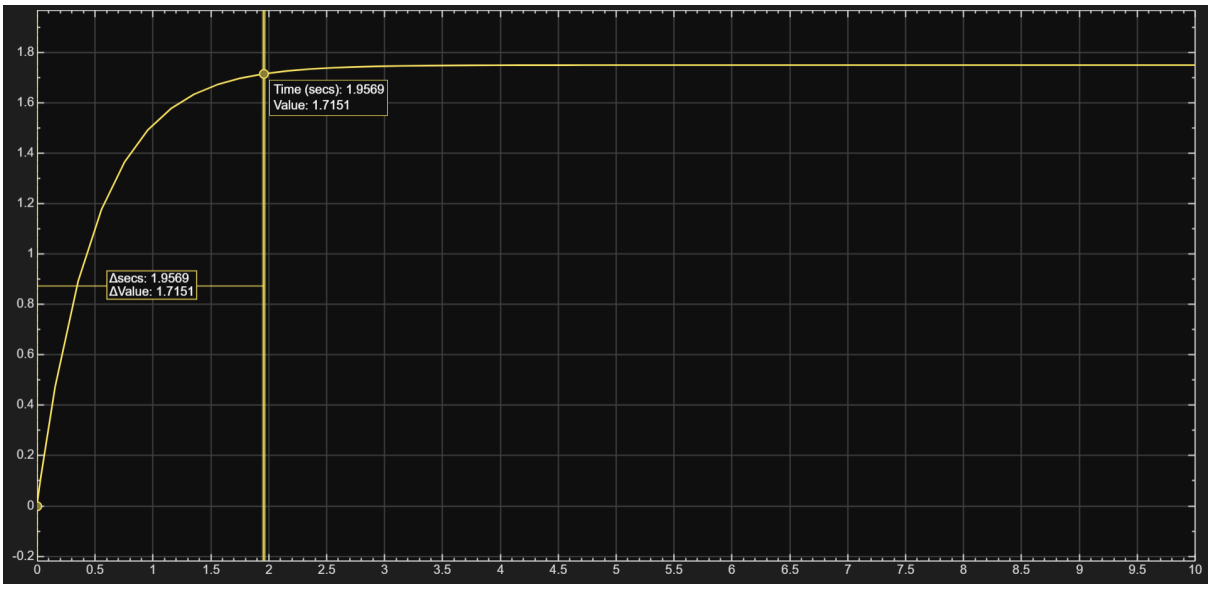

Figura 6: Gráfica obtenida en Simulink mostrando el tiempo de establecimiento con entrada de escalón unitario.

- **Tiempo de Subida**
  El tiempo de subida (aquel en que pasa del 10% al 90% del valor final) fue: **1.0968 s**

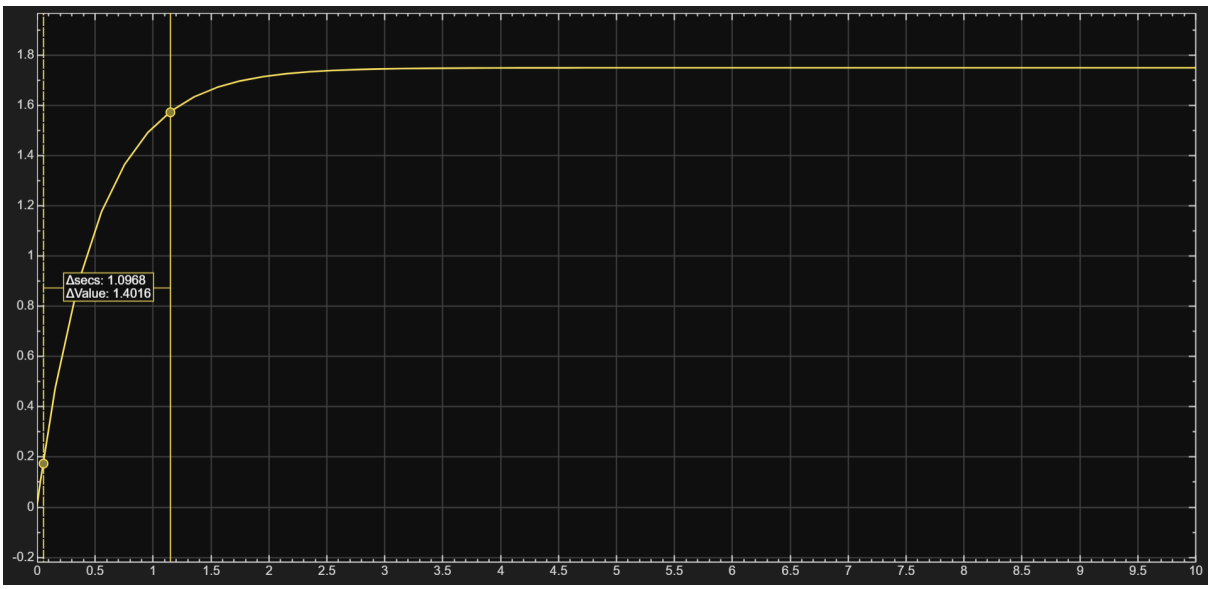

Figura 7: Gráfica obtenida en Simulink mostrando el tiempo de subida con entrada de escalón unitario.

- **Forma de la curva**: exponencial

Experimento 2. Respuesta a escalón de amplitud 5
Se analizaron los siguientes parámetros:

- **Proporcionalidad**
  Al aplicar un escalón de amplitud 5, la salida final alcanza aproximadamente 8.75, cumpliendo la relación

$$y_{ss} = K \cdot u$$

  Esto demuestra que la salida escala directamente con la entrada. Se confirma así el comportamiento lineal del sistema.

- **Rapidez**
  El tiempo de establecimiento permanece prácticamente igual que en el escalón unitario, ya que depende únicamente de la constante de tiempo τ. La amplitud de la entrada no afecta la rapidez del sistema. Esto es característico de sistemas de primer orden lineales.

- **Comportamiento dinámico**
  La respuesta conserva su forma exponencial creciente sin presentar sobre impulso ni oscilaciones. El sistema sigue siendo estable y converge suavemente al valor final. Su dinámica no cambia ante variaciones en la magnitud de la entrada.

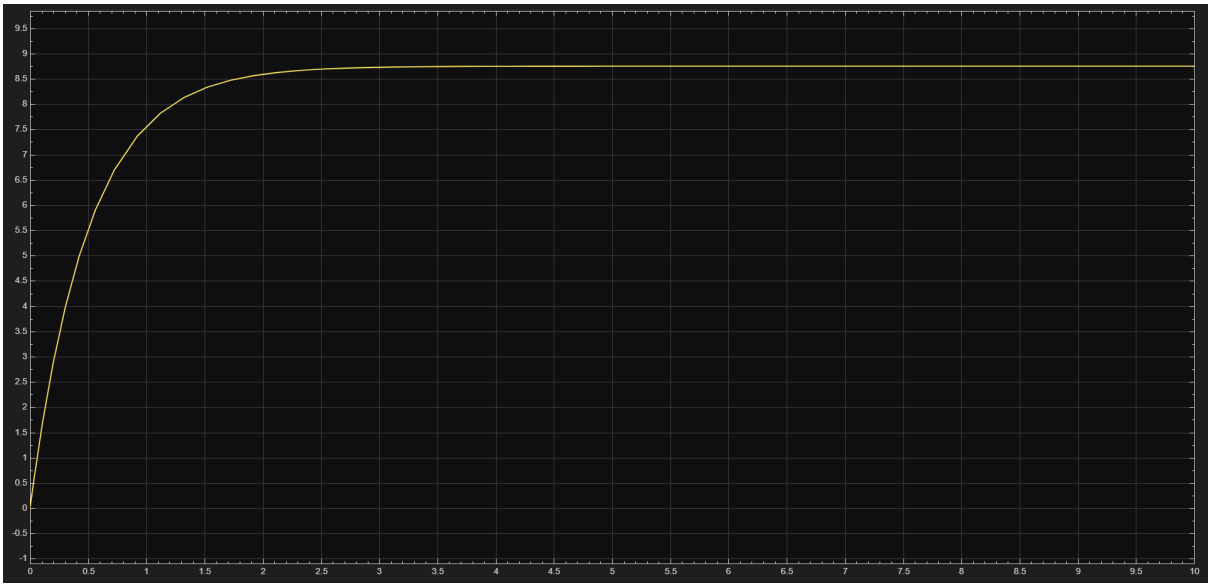

Figura 8: Gráfica obtenida en Simulink mostrando la salida para una entrada de escalón con amplitud 5.

Por otro lado, al observar, simular y analizar el sistema podemos responder a las siguientes preguntas, además de haber realizado previamente la comparación entre los valores teóricos y simulados:

¿Qué tipo de sistema es? (orden, estabilidad, tipo)

- **Orden:** Primer orden
- **Estabilidad:** Estable, esto debido a que si se factoriza el denominador para obtener el polo obtenemos que

  $$0.5(s + 2)$$

  por lo tanto:

  $$s = -2$$

  Lo cual significa que no hay polo en el origen y no hay integrador lo cual lo hace estable
- **Tipo:** Tipo 0, ya que su función de transferencia no presenta polos en el origen ni integradores puros. Esto implica que, en lazo cerrado, puede presentar error en estado estacionario ante entradas tipo escalón si no se incorpora acción integral

Preparación para PID

Para preparación de PID sin diseñar PID aún, podemos responder a las siguientes preguntas:

1. **¿Qué problema tiene el motor sin control?**
   El sistema en lazo abierto no puede corregir errores respecto a una referencia deseada. Si existe perturbación o cambio en parámetros, el sistema no lo compensa.

2. **¿Seguiría bien una referencia variable?**
   No. Debido a su constante de tiempo, siempre presentaría retraso y posible error en seguimiento dinámico.

3. **¿Por qué se necesita PID?**
   Esto es ya que reduce el error en estado estacionario, mejora la rapidez y aumenta la estabilidad ante cambios bruscos o cambios de referencia

4. **¿Qué esperas que haga el término P?**
   El término P se espera que reduzca el error actual aumentando la rapidez de respuesta.

5. **¿Qué hará el término I?**
   El término I elimina el error en estado estacionario acumulando el error en el tiempo como una predicción

Resultados

Para resolver este problema se decidió utilizar el auto tuning de PID que es una herramienta de matlab que facilita el encontrar los parámetros o el ajuste correspondiente al controlador.

Modelado en Simulink (Lazo Cerrado)
Se implementó el modelo continuo utilizando:

- Bloque Step
- Bloque Transfer Function
- Bloque Scope
- Bloque de suma
- Bloque de PID

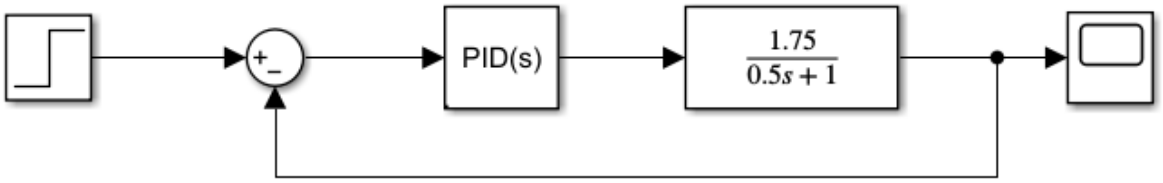

Figura 9: Diagrama en Simulink del sistema en lazo cerrado.

De este modo se hizo la sintonización de los parámetros utilizando la herramienta de MATLAB & Simulink Control Systems el cual, cuando se modela el comportamiento de un sistema a lazo cerrado con control PID, se puede hacer un **self-tuning** a las ganancias del control. En su interfaz gráfica se puede modificar qué tan agresivo es el controlador junto con el tiempo de respuesta deseado con una gráfica que indica cómo se comportaría moviendo tales parámetros y se puede ir ajustando con la referencia del sistema de entrada.

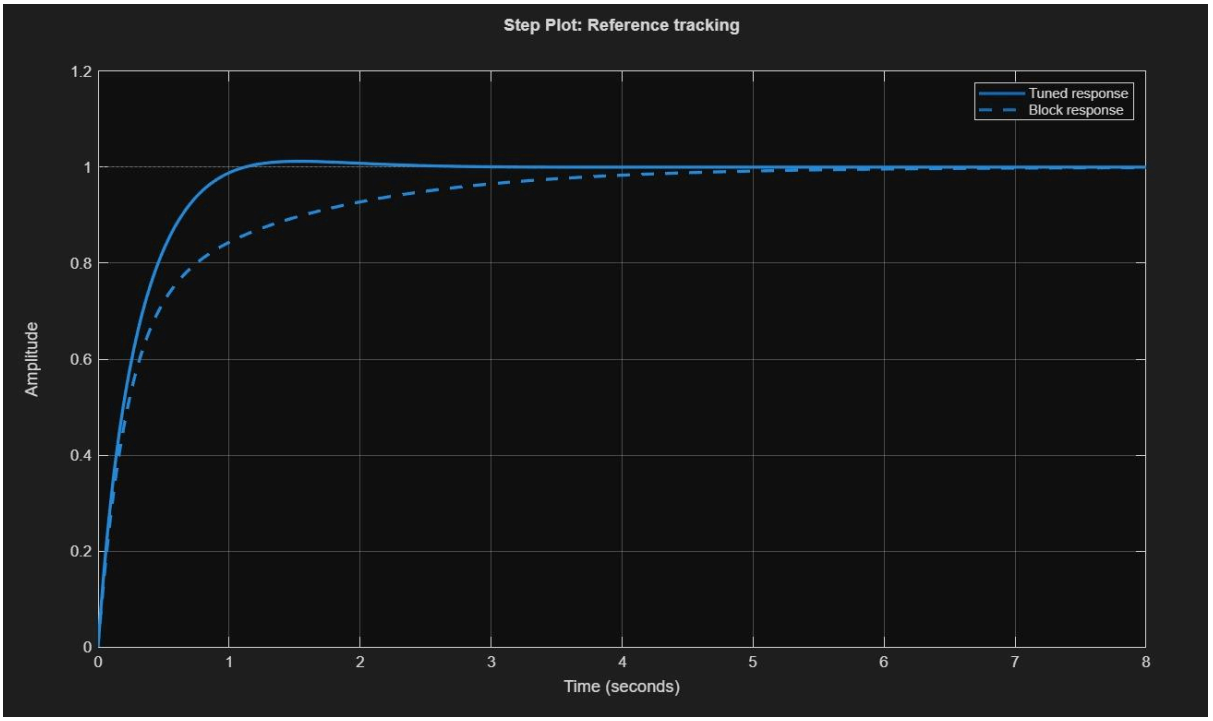

Figura 10: Respuesta en lazo abierto al escalón unitario vs. Respuesta en lazo cerrado con el controlador PID.

Una vez encontrado el punto de estabilidad del sistema, la herramienta ajustará los parámetros del controlador automáticamente. Para este sistema, como se observa en la figura 11, se obtuvo un controlador de tipo PID.

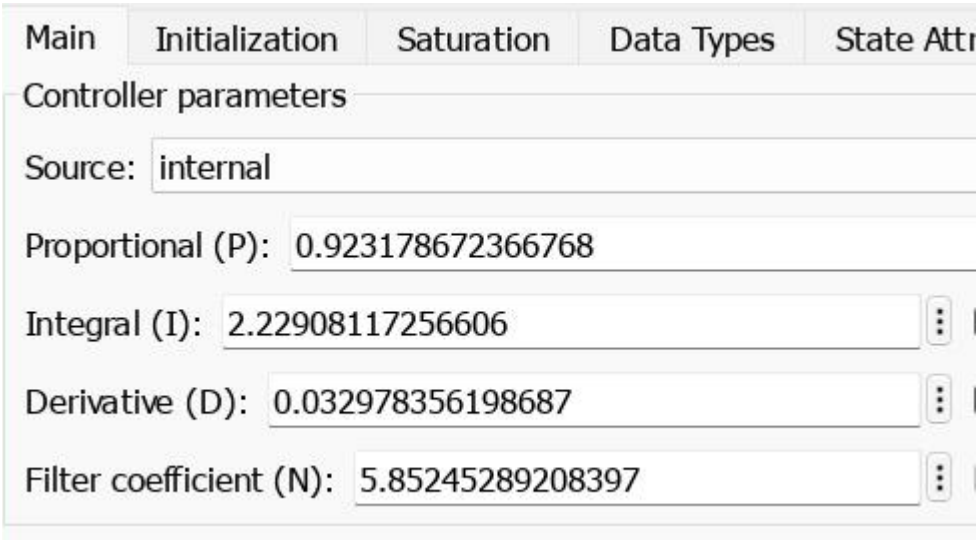

Figura 11: Sintonización automática de las ganancias del controlador.

Conclusiones

Se logró modelar correctamente el motor y comprobar que corresponde a un sistema de primer orden estable con constante de tiempo $\tau = 0.5$ s y ganancia $K = 1.75$. El análisis

experimental coincidió con el análisis teórico tanto en valor final como en tiempo de establecimiento. Así mismo se comprobó que el sistema en lazo abierto presenta limitaciones importantes para el seguimiento de referencia, lo que justifica la implementación de un controlador PID. La sintonización permitió mejorar la rapidez y eliminar el error en estado estacionario, cumpliendo los objetivos planteados.

Como mejora futura, podría implementarse un análisis más formal utilizando Ziegler-Nichols o métodos de optimización automática para comparar desempeño.

Referencias

Åström, K., & Hägglund, T. (2004). Revisiting the Ziegler–Nichols step response method for PID control. *Journal Of Process Control, 14*(6), 635-650. https://doi.org/10.1016/j.jprocont.2004.01.002

MathWorks. (s. f.-a). *PID Tuner*. MATLAB Help Center. Recuperado 23 de febrero de 2026, de https://www.mathworks.com/help/control/ref/pidtuner-app.html

MathWorks. (s. f.-b). *PID Tuning Algorithm*. MATLAB Help Center. Recuperado 23 de febrero de 2026, de https://www.mathworks.com/help/control/getstart/pid-tuning-algorithm.html

Vilanova, R., & Alfaro, V. M. (2011). Control PID robusto: Una visión panorámica. *Revista Iberoamericana de Automática E Informática Industrial RIAI, 8*(3), 141-158. https://doi.org/10.1016/j.riai.2011.06.003