Universidad Miguel Hernández de Elche

MASTER UNIVERSITARIO EN ROBÓTICA

“Desarrollo de una aplicación interactiva para el diseño de controladores PID basada en Python”

Trabajo de Fin de Máster

Curso académico 2022/2023

Autor: Luis Alfonso Riera Abellán
Tutor: Eduardo Iáñez Martínez

IAÑEZ MARTINEZ EDUARDO - 48359384K

Firmado digitalmente por IAÑEZ MARTINEZ EDUARDO - 48359384K
Fecha: 2023.06.28 16:08:15 +02'00'

Agradecimientos:

Quisiera agradecer este proyecto a mi tutor *Eduardo Iáñez Martínez* por brindarme la oportunidad de realizar un proyecto tan interesante. Mi más sincero agradecimiento por la orientación y el asesoramiento que me ha proporcionado durante el transcurso del proyecto.

También quisiera agradecer todo el apoyo emocional y la motivación que me han otorgado mis seres queridos, sin ellos no hubiera sido posible superar todas las dificultades que han aparecido en este proyecto y en toda esta etapa universitaria.

Me siento afortunado de haber podido cursar este máster universitario tan novedoso y quisiera agradecer al comité de admisión del máster el haberme dado la oportunidad de poder cursarlo.

Por último, doy las gracias a los profesores de este máster por su dedicación, conocimiento y paciencia. Su guía y apoyo han sido fundamentales para mi crecimiento académico y personal

Índice

1. Introducción 8

1.1. Resumen 8

1.2. Objetivos 8

1.3. Contextualización 8

1.4. Aplicaciones 8

1.5. Estructura de la memoria 9

1.5.1. Introducción 9

1.5.2. Estado del Arte 9

1.5.3. Materiales y métodos 9

1.5.4. Resultados 9

1.5.5. Conclusiones 9

2. Estado del Arte 10

2.1. Conceptos fundamentales de la teoría de control 10

2.1.1. Sistema de control en bucle cerrado 10

2.1.2. Sistema de control en bucle abierto 11

2.1.3. Realimentación 12

2.1.4. Error de Control 13

2.2. Control PID 14

2.2.1. Acciones de Control 14

2.2.2. Selección de valores de Ganancia 15

2.2.3. Parámetros del controlador y requisitos del sistema 16

2.3. Métodos de diseño y sintonización 17

2.3.1. Método de Ziegler-Nichols: 19

2.3.2. Método de Cohen-Coon: 21

2.3.3. Método de Tyreus-Luyben: 22

2.3.4. Método de IMC (Internal Model Control): 22

2.3.5. Método básico basado LDR: 23

2.3.6. Método respuesta en frecuencia: 25

2.3.7. Método algoritmos genéticos 26

2.3.8. Método LQR 27

2.4. Entornos para la implementación y diseño de controladores 28

2.4.1. Matlab y Simulink 28

2.4.2. Python 29

2.4.3. C/C++: 29
2.4.4. LabVIEW: 30
2.5. Aplicaciones controladores PID 31
2.5.1. Controladores PID en la robótica 31
3. Material y métodos 33
3.1. Herramientas 33
3.1.1. Jupyter Notebook: 33
3.1.2. Google Colab: 33
3.1.3. Librerías: 33
3.2. Funciones 34
3.2.1. Llamada de funciones entre cuadernos 34
3.2.2. Respuesta de sistemas 38
3.2.3. Representación en el plano complejo 43
3.2.4. Parámetros 47
3.3. Implementación métodos diseño PID 48
3.3.1. Método Manual 48
3.3.2. Método fuerza bruta 49
3.3.3. Método del Lugar de las Raíces 51
3.3.4. Método de respuesta en frecuencia 53
3.4. Otros cuadernos de análisis de sistemas 55
4. Resultados 57
4.1. Método Manual 57
4.2. Métodos de diseño 58
4.2.1. Método fuerza bruta 58
4.2.2. Método del lugar de las raíces 59
4.2.3. Método de respuesta en frecuencia 61
4.3. Método de Ziegler-Nichols 61
4.3.1. Caso 1 61
4.3.2. Caso 2 62
5. Conclusiones 63
5.1. Conclusiones 63
5.2. Trabajos futuros 64
6. Bibliografía 66

Ilustraciones y tablas

Ilustración 1. Diagrama sistema bucle cerrado 10
Ilustración 2. Diagrama sistema bucle abierto 11
Ilustración 3. Feedback negativo 13
Ilustración 4. Parámetros Controlador 15
Ilustración 5. Diagrama bloques IMC 23
Ilustración 6. LDR en función valores k 24
Ilustración 7. Ejemplo esquema control temperatura 31
Ilustración 8. Ejemplo definir funciones 34
Ilustración 9. Descargar .py 35
Ilustración 10. Ejemplo completo importar cuadernos 35
Ilustración 11. Atajo copiar Ruta 36
Ilustración 12. Desmontar 37
Ilustración 13. Revocar 37
Ilustración 14. ver_respuesta_escalon 38
Ilustración 15. respuesta_temporal 38
Ilustración 16. Respuesta_temporal_tmanual 39
Ilustración 17. Respuesta_temporal_2 39
Ilustración 18. Respuesta_temporal_2 (y) 40
Ilustración 19. tipo_respuesta_2do_orden 40
Ilustración 20. pintar_respuesta_1er_orden 41
Ilustración 21. pintar_respuesta_2do_orden_subamortiguado 41
Ilustración 22. pintar_respuesta_2do_orden_subamortiguado_simplificado 42
Ilustración 23. pintar_respuesta_2do_orden_crit_amort 42
Ilustración 24. pintar_respuesta_2do_orden_sobreamortiguado 43
Ilustración 25. Funciones Puntar_G, pintar_polos y pintar_ejes 46
Ilustración 26. LDR Automático 46
Ilustración 27. LDR Manual en función de k 55
Ilustración 28. Regiones 56
Ilustración 29. Ejemplo datos salida método manual 57
Ilustración 30. Gráfica de salida del sistema compensado con método manual 57
Ilustración 31. Ejemplo valorar_caso_LDR 60
Ilustración 32. Controlador con LDR 60
Ilustración 33. Respuesta del sistema controlado 60
Ilustración 34. Diagramas de bode 61
Ilustración 35. Ejemplo aplicación Ziegler-Nichols I 61
Ilustración 36. Ziegler Nichols II 62

Tabla 1. Ziegler-Nichols I 20
Tabla 2. Ziegler Nichols II 21
Tabla 3. Cohen-Coon 22
Tabla 4. Tyreus-Luyben 22
Tabla 5. Ejemplo tabla de resultados método fuerza bruta 58

1. Introducción

1.1. Resumen

En el presente trabajo se pretende desarrollar un conjunto de herramientas interactivas que permitan el análisis de sistemas continuos y diferentes métodos para el diseño de controladores PID en Jupyter Notebook bajo Python. Dichas herramientas permitirán visualizar la información del sistema, así como del sistema junto al controlador diseñado para comprobar que cumple con los requisitos planteados y poder realizar ajustes sobre éste.

1.2. Objetivos

El objetivo de este trabajo será el desarrollo de un conjunto de herramientas en Jupyter Notebook bajo python. Esta aplicación permitirá el análisis de sistemas continuos y diferentes métodos para el diseño de controladores PID según las condiciones solicitadas. Las herramientas serán interactivas permitiendo visualizar la información del sistema, así como del sistema junto al controlador diseñado para comprobar que cumple con los requisitos planteados y poder realizar ajustes sobre éste.

1.3. Contextualización

Este proyecto surge como una propuesta innovadora gracias a las ventajas que nos puede ofrecer la estructuración de programas en forma de cuadernos (Jupyter Notebook) como la accesibilidad en la nube, los recursos computacionales escalables, la integración con Google Drive, librerías y entornos preinstalados, la facilidad de colaboración y compartición y la interfaz interactiva.

1.4. Aplicaciones

Este proyecto ha sido desarrollado con el fin principal de poder ser empleado como herramienta para el aprendizaje, la experimentación y la mejora de los sistemas de control.

1.5. Estructura de la memoria

1.5.1. Introducción

Se explica de manera clara y concisa en qué consiste este proyecto, cuáles son los objetivos del mismo y la estructura de presente la memoria…

1.5.2. Estado del Arte

En este capítulo de la memoria se realiza un estudio profundo de los sistemas continuos y de los diferentes métodos de diseño de controladores PID. Esta introducción teórica ayuda a entender todas las herramientas desarrolladas durante el proyecto.

1.5.3. Materiales y métodos

Se explica el entorno de programación seleccionado para realizar dicha aplicación interactiva y las diferentes librerías que se han empleado para cubrir los objetivos propuestos.

1.5.4. Resultados

Aquí se recogen las herramientas implementadas en este proyecto, teniendo en cuenta el estudio teórico realizado en el estado del arte para determinar cuáles son las herramientas más eficaces para el análisis de sistemas continuos y para el diseño de controladores PID.

1.5.5. Conclusiones

En el siguiente capítulo se exponen las conclusiones obtenidas tras la realización del proyecto, así como las dificultades encontradas durante su ejecución y los posibles trabajos o herramientas futuras que se podrían desarrollar.

2. Estado del Arte

Para comprender adecuadamente los diferentes métodos de diseño de controladores PID, es importante tener una comprensión sólida de los siguientes aspectos del estado del arte en teoría de control.

2.1. Conceptos fundamentales de la teoría de control

Antes de profundizar en los métodos de diseño de controladores PID, es importante comprender los conceptos fundamentales de la teoría de control, como los sistemas de control en bucle cerrado, los sistemas de control en bucle abierto, la retroalimentación y el error de control.

2.1.1. Sistema de control en bucle cerrado

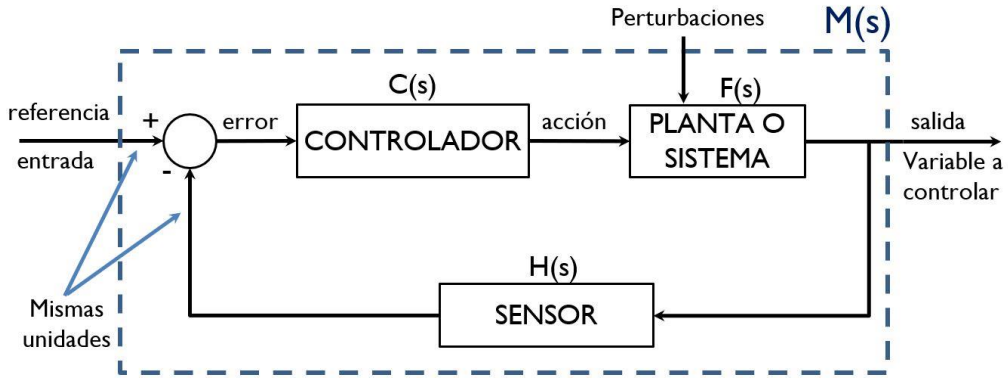

Ilustración 1. Diagrama sistema bucle cerrado

Un sistema de control en bucle cerrado, también conocido como sistema de control realimentado, es un tipo de sistema de control que utiliza la retroalimentación para medir y corregir la salida del sistema en función de una referencia deseada. En otras palabras, el sistema de control en bucle cerrado compara la salida del sistema Y(s) con una señal de referencia R(s) y utiliza la diferencia entre ambas para ajustar el control del sistema y mantener la salida lo más cerca posible de la referencia.

El esquema sistema de control en bucle cerrado (mostrado en la Ilustración 1), se compone de cuatro elementos principales: la planta, el sensor, el controlador y la señal de referencia. La planta es el sistema que se está controlando, como un

motor, un proceso químico o un sistema de climatización. El sensor mide la salida de la planta y la compara con la señal de referencia, generando una señal de error.

El controlador utiliza la señal de error para ajustar el control de la planta, utilizando una realimentación que se aplica a la entrada de la planta.

Es importante explicar aquello que denominamos función de transferencia del sistema, según OGATA, K. (1998). "la función de transferencia de un sistema descrito mediante una ecuación diferencial lineal e invariante con el tiempo se define como el cociente entre la transformada de Laplace de la salida (función de respuesta) y la transformada de Laplace de la entrada (función de excitación) bajo la suposición de que todas las condiciones iniciales son cero." (INGENIERIA DE CONTROL MODERNA (3a. ed.)) (p. 60) [1].

El sistema de control en bucle cerrado tiene varias ventajas en comparación con el sistema de control en bucle abierto. En primer lugar, el sistema de control en bucle cerrado es más preciso y estable, ya que utiliza la retroalimentación para corregir cualquier error en la salida del sistema. En segundo lugar, el sistema de control en bucle cerrado es más resistente a las perturbaciones externas, ya que puede ajustar el control del sistema para compensar los cambios en las condiciones externas. En tercer lugar, el sistema de control en bucle cerrado es más flexible, ya que puede ajustar el control del sistema para alcanzar diferentes objetivos, como minimizar el consumo de energía o maximizar la eficiencia.

La implementación de un sistema de control en bucle cerrado implica la selección cuidadosa de los componentes y la configuración adecuada de los parámetros del sistema. La selección del sensor y del controlador debe realizarse en función de las características de la planta y del objetivo del sistema de control. La sintonización del controlador es un proceso crítico que implica la selección de los valores adecuados de las constantes proporcional, integral y derivativa, que ajustan el comportamiento del controlador para lograr un rendimiento óptimo.

2.1.2. Sistema de control en bucle abierto

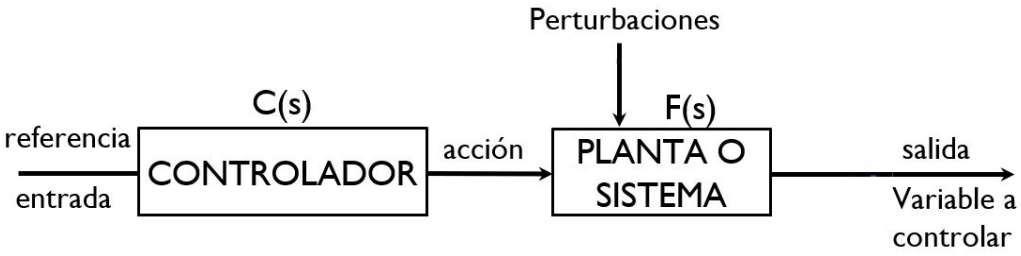

Ilustración 2. Diagrama sistema bucle abierto

Un sistema de control en bucle abierto (esquema representado en la Ilustración 2), también conocido como sistema de control de lazo abierto, es un tipo de sistema de control en el que la salida del sistema no se mide ni se compara con una

referencia deseada. En lugar de eso, el sistema se controla utilizando un conjunto predefinido de entradas y una función de transferencia conocida que relaciona las entradas y las salidas del sistema.

El sistema de control en bucle abierto se compone de tres elementos principales: la planta, la entrada y la función de transferencia. La planta es el sistema que se está controlando, como un motor, un proceso químico o un sistema de climatización. La entrada es la señal que se envía a la planta, y la función de transferencia describe la relación matemática entre la entrada y la salida del sistema.

El sistema de control en bucle abierto tiene algunas limitaciones en comparación con el sistema de control en bucle cerrado. En primer lugar, el sistema de control en bucle abierto no puede corregir ningún error en la salida del sistema, ya que no hay una señal de retroalimentación para comparar la salida con una referencia deseada. En segundo lugar, el sistema de control en bucle abierto es más susceptible a las perturbaciones externas, ya que no puede ajustar el control del sistema para compensar los cambios en las condiciones externas. En tercer lugar, el sistema de control en bucle abierto no es tan flexible como el sistema de control en bucle cerrado, ya que solo puede realizar una tarea específica para la cual fue diseñado. [1] y [2]

La implementación de un sistema de control en bucle abierto implica la selección cuidadosa de los componentes y la configuración adecuada de los parámetros del sistema. La selección de la función de transferencia debe basarse en las características de la planta y en los objetivos del sistema de control. Además, la selección de la entrada debe ser adecuada para el tipo de planta y para el resultado deseado del sistema de control.

2.1.3. Realimentación

La realimentación, también conocida como retroalimentación o feedback, es un concepto fundamental en la teoría de control. En esencia, la retroalimentación implica tomar una medida de la salida de un sistema y utilizar esta información para ajustar la entrada del sistema de tal manera que la salida se acerque a una referencia deseada.

La realimentación se utiliza en sistemas de control en bucle cerrado, donde la salida del sistema se mide y se compara con una referencia deseada, y esta diferencia se utiliza para ajustar la entrada del sistema. La retroalimentación puede ser positiva o negativa, dependiendo de si la entrada se ajusta para amplificar o reducir la diferencia entre la salida y la referencia deseada.

En la Ilustración 3, se muestra el feedback negativo, que es el tipo más común de retroalimentación utilizada en sistemas de control en bucle cerrado. En este caso,

la señal de retroalimentación se utiliza para reducir la diferencia entre la salida y la referencia deseada, lo que lleva a una mayor precisión y estabilidad del sistema.

La retroalimentación positiva, por otro lado, amplifica la diferencia entre la salida y la referencia deseada, lo que puede llevar a inestabilidades y oscilaciones en el sistema.

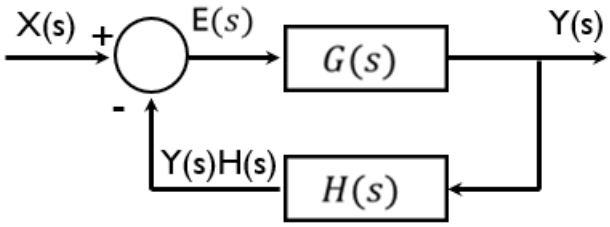

Ilustración 3. Feedback negativo

La retroalimentación se utiliza en una variedad de aplicaciones, desde el control de procesos industriales hasta la estabilización de vehículos y la automatización de sistemas eléctricos y electrónicos. La retroalimentación también se utiliza en la teoría de sistemas y la teoría de la información para medir y mejorar el rendimiento de los sistemas.

En resumen, la retroalimentación es un concepto fundamental en la teoría de control que implica utilizar una medida de la salida de un sistema para ajustar la entrada del sistema de tal manera que la salida se acerque a una referencia deseada. La retroalimentación puede ser positiva o negativa, dependiendo de si la entrada se ajusta para amplificar o reducir la diferencia entre la salida y la referencia deseada. La realimentación se utiliza en una variedad de aplicaciones y es esencial para la precisión y la estabilidad de los sistemas de control en bucle cerrado.

2.1.4. Error de Control

El error de control es una medida de la discrepancia entre la salida real de un sistema controlado y su valor deseado o de referencia. En teoría de control, el objetivo es reducir este error tanto como sea posible mediante el diseño y la implementación de un controlador adecuado.

Existen varios tipos de errores de control que pueden ocurrir, dependiendo de la forma en que se define la referencia y se mide la salida del sistema. El error de seguimiento es una medida de la diferencia entre la salida real del sistema y el valor deseado o de referencia. El error de seguimiento puede ser debido a una variedad de factores, como errores de medición, perturbaciones externas, imprecisiones en el modelo matemático del sistema, y errores en la implementación del controlador.

El error en estado estacionario es una medida de la discrepancia entre la salida real del sistema y su valor deseado cuando el sistema se ha estabilizado y se ha alcanzado un equilibrio dinámico. El error en estado estacionario puede ser debido a la presencia de un sesgo o error sistemático en el sistema, o a limitaciones en el controlador que impiden que la salida alcance el valor deseado. El error en régimen permanente queda definido por la siguiente fórmula:

$$ep = \lim_{s \to 0} \frac{sX(s)}{1 + G(s)H(s)}$$

El error transitorio es una medida de la discrepancia entre la salida real del sistema y su valor deseado durante el periodo de tiempo en que el sistema se está estabilizando después de una perturbación o cambio en la entrada. El error transitorio puede ser debido a la presencia de factores como la inercia del sistema, los efectos de los retardos en el tiempo, y los efectos de los filtros de respuesta limitada.

El objetivo del diseño de un controlador es minimizar estos tipos de errores de control y mantener la salida del sistema lo más cerca posible del valor deseado o de referencia. El controlador debe ser diseñado para ser lo suficientemente robusto y adaptable para manejar las perturbaciones y cambios en el sistema que puedan afectar el rendimiento del sistema.

En resumen, el error de control es una medida de la discrepancia entre la salida real de un sistema controlado y su valor deseado o de referencia. Existen varios tipos de errores de control que pueden ocurrir, incluyendo el error de seguimiento, el error en estado estacionario y el error transitorio. El diseño de un controlador adecuado debe tener como objetivo minimizar estos errores y mantener la salida del sistema lo más cerca posible del valor deseado o de referencia.

2.2. Control PID

Es importante conocer los principios básicos del controlador PID, incluyendo las tres acciones de control (proporcional, integral y derivativa), la selección de los valores de ganancia y la configuración de los parámetros del controlador.

2.2.1. Acciones de Control

El control PID (Proporcional-Integral-Derivativo) es uno de los tipos de controladores más utilizados en la industria debido a su simplicidad y eficacia en el control de sistemas dinámicos. El controlador PID se basa en tres acciones de

control principales: proporcional, integral y derivativa, que trabajan juntas para reducir el error de control.

El principio básico del controlador PID es comparar la salida real del sistema con un valor deseado o de referencia, y utilizar esta diferencia (el error) para ajustar la entrada del sistema de manera que la salida se acerque lo más posible al valor deseado. El controlador PID utiliza tres términos para realizar este ajuste: proporcional, integral y derivativo.

El término proporcional ajusta la entrada del sistema en proporción al error actual. Esto significa que cuanto mayor sea el error, mayor será el ajuste realizado por el término proporcional. La acción proporcional es útil para reducir el error de seguimiento, pero no es capaz de corregir el error en estado estacionario.

El término integral ajusta la entrada del sistema en función del error acumulado a lo largo del tiempo. Esto significa que cuanto más tiempo el error esté presente, mayor será el ajuste realizado por el término integral. La acción integral es útil para corregir el error en estado estacionario, pero puede introducir oscilaciones en el sistema si se utiliza incorrectamente.

El término derivativo ajusta la entrada del sistema en función de la tasa de cambio del error. Esto significa que cuanto más rápido cambie el error, mayor será el ajuste realizado por el término derivativo. La acción derivativa es útil para reducir las oscilaciones en el sistema, pero puede amplificar el ruido y la perturbación del sistema si se utiliza incorrectamente. [3]

2.2.2. Selección de valores de Ganancia

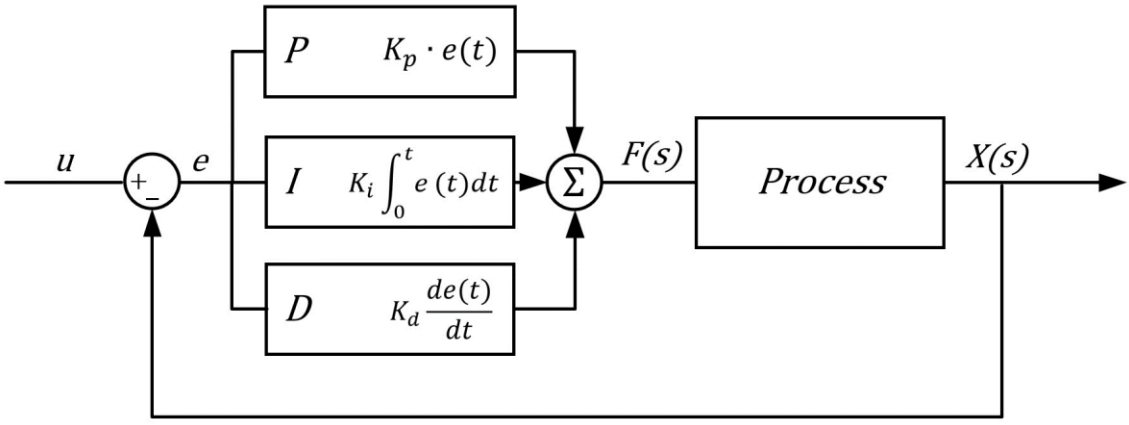

Ilustración 4. Parámetros Controlador

La combinación de estos tres términos (Ilustración 4) en el controlador PID permite ajustar la entrada del sistema de manera que se reduzca el error de control. Sin embargo, la selección adecuada de los valores de ganancia y la configuración de los parámetros del controlador es esencial para el éxito del controlador PID. [3]

La ganancia proporcional (Kp) determina la magnitud del ajuste realizado por el término proporcional. Una ganancia proporcional alta puede reducir el error de seguimiento, pero puede introducir oscilaciones en el sistema. Una ganancia proporcional baja puede reducir las oscilaciones en el sistema, pero puede no ser capaz de reducir el error de seguimiento.

La ganancia integral (Ki) determina la magnitud del ajuste realizado por el término integral en función del tiempo acumulado. Una ganancia integral alta puede corregir el error en estado estacionario, pero puede introducir oscilaciones en el sistema. Una ganancia integral baja puede reducir las oscilaciones en el sistema, pero puede no ser capaz de corregir el error en estado estacionario.

La ganancia derivativa (Kd) determina la magnitud del ajuste realizado por el término derivativo en función de la tasa de cambio del error. Una ganancia derivativa alta puede reducir las oscilaciones en el sistema, pero puede amplificar el ruido y la perturbación del sistema. Una ganancia derivativa baja puede no ser capaz de reducir las oscilaciones en el sistema.

2.2.3. Parámetros del controlador y requisitos del sistema

La configuración de los parámetros del controlador también es importante para el rendimiento del controlador PID. Es importante destacar que la selección de los valores de ganancia y la configuración de los parámetros del controlador dependen en gran medida de las características del sistema a controlar. Por lo tanto, se recomienda realizar una evaluación cuidadosa del sistema antes de configurar el controlador PID.

Entre los parámetros que deben configurarse se encuentran la frecuencia de muestreo, la escala de tiempo y la elección del tipo de controlador (por ejemplo, si se utiliza un controlador PID clásico o un controlador PID de orden fraccional). Además, es importante prestar atención a la capacidad de cálculo del sistema de control, ya que algunos métodos de configuración de parámetros pueden ser computacionalmente costosos.

En general, la configuración adecuada de los parámetros del controlador PID es crucial para lograr un buen rendimiento del sistema de control y garantizar su estabilidad y fiabilidad.

A continuación, se describen algunos de los parámetros que se ajustan derivados de los valores de ganancia de los controladores PID:

- Tiempo integral (Ti): Controla el tiempo en que el controlador integra la señal de error. Un valor más alto de Ti permite que el controlador integre la señal de error durante un período de tiempo más largo, lo que puede aumentar la precisión en estado estacionario, pero también puede aumentar el tiempo de respuesta y la sobreoscilación.

- Tiempo derivativo (Td): Controla el tiempo en que el controlador deriva la señal de error. Un valor más alto de Td permite que el controlador derive la señal de error durante un período de tiempo más largo, lo que puede reducir la sobreoscilación y mejorar la estabilidad, pero también puede aumentar la posibilidad de ruido y aumentar el tiempo de respuesta.
- K_R o K_p es la ya descrita ganancia de la acción proporcional.

$$Gr(s) = \frac{U(s)}{E(s)} = Kr(1 + \frac{1}{Ti * s} + Td * s)$$

Las especificaciones de un sistema de control son los requisitos y características que se establecen para determinar cómo debe funcionar el controlador en un sistema de control. Estas especificaciones ayudan a definir los parámetros del controlador que se diseñará.

Las características operativas de un sistema de regulación engloban el conjunto de atributos que deben presentar dicho sistema para cumplir con objetivos específicos de funcionamiento.

Los sistemas de control son diseñados para llevar a cabo tareas particulares. Para especificar los requisitos que el sistema de control debe cumplir, se utilizan las características operativas requeridas. Estas características suelen abarcar la estabilidad, la precisión y la velocidad de respuesta.

A menudo, las características operativas de los sistemas de regulación se describen en relación a un sistema de segundo orden normalizado. Cuanto más se asemeje el sistema tratado a un sistema de segundo orden, mayor será la precisión con la que se cumplirán las características requeridas.

Las características más comúnmente utilizadas son: las constantes de error Kp, Kv, Ka, el tiempo de subida tr, el tiempo de establecimiento ts, el tiempo de pico tp, el porcentaje de sobreoscilación M_p, ancho de banda WB, frecuencia de resonancia wr, margen de fase γ, frecuencia natural wn, factor de amortiguamiento ξ…

Estas características no son completamente independientes entre sí. De hecho, si se conocen algunas de ellas, no es necesario proporcionar las demás, ya que contienen la misma información.

2.3. Métodos de diseño y sintonización

El diseño de controladores PID y la sintonización de controladores PID son procesos diferentes en la ingeniería de control. El diseño de controladores PID implica la selección de los parámetros del controlador PID (proceso de ajuste o configuración), lo que incluye elegir los valores adecuados para las constantes proporcional, integral y derivativa (Kp, Ki, Kd), así como el tipo de estructura del

controlador (en serie, en paralelo, etc.) para obtener un controlador que proporcione una respuesta adecuada y satisfactoria en términos de rendimiento y estabilidad para el sistema de control.

Los métodos de diseño se enfocan en establecer los valores óptimos para los parámetros del controlador que permitan obtener una respuesta óptima del sistema, según los requisitos de desempeño específicos.

Por otro lado, la sintonización de controladores PID es el proceso de ajustar los parámetros del controlador PID para lograr un desempeño óptimo del sistema de control en términos de estabilidad, precisión, rapidez de respuesta, amortiguamiento de oscilaciones, entre otros. La sintonización de un controlador PID se refiere al ajuste de los parámetros del controlador PID en tiempo real, para lograr un buen desempeño del sistema en presencia de perturbaciones o cambios en las condiciones de operación. Los métodos de sintonización se centran en el ajuste de los parámetros del controlador para mejorar el comportamiento del sistema ante condiciones variables, minimizar el tiempo de respuesta y reducir la sobreoscilación o el error en estado estacionario.

Es decir, el diseño de un controlador PID implica seleccionar la estructura y los parámetros del controlador, mientras que la sintonización de un controlador PID implica ajustar los parámetros del controlador ya diseñado para lograr un mejor desempeño del sistema de control.

En resumen, el diseño de un controlador PID se realiza antes de la implementación y se enfoca en seleccionar valores óptimos para los parámetros del controlador, mientras que la sintonización de un controlador PID se realiza después de la implementación y se enfoca en ajustar los parámetros del controlador para optimizar el desempeño del sistema en tiempo real.

Para seleccionar los valores de las constantes Kp, Ki y Kd, se pueden utilizar criterios de diseño como el método básico basado en el lugar de las raíces, el método de la respuesta en frecuencia o el método de Ziegler-Nichols entre otros. Estos métodos permiten seleccionar los valores de Kp, Ki y Kd en función de los requisitos de rendimiento y estabilidad del sistema de control, sin la necesidad de ajustarlos posteriormente a través de la sintonización.

También se pueden utilizar herramientas de diseño asistido por computador, como simulaciones de sistemas de control, para evaluar el desempeño del controlador PID diseñado antes de implementarlo en el sistema real. De esta forma, se pueden hacer ajustes en el diseño antes de implementarlo en el sistema de control.

El diseño de controladores PID es un proceso importante en la automatización de sistemas y procesos. En general, hay dos enfoques principales para el diseño de controladores PID: basado en la sintonización y basado en el diseño.

El enfoque basado en la sintonización implica ajustar los parámetros del controlador en función de las respuestas del sistema real. Este enfoque puede ser muy efectivo para sistemas complejos donde no se dispone de un modelo

matemático preciso del sistema. Sin embargo, la sintonización puede ser un proceso costoso y lento, y puede no garantizar un rendimiento óptimo.

A continuación, se describen algunas de las formas más comunes de sintonización de controladores PID:

2.3.1. Método de Ziegler-Nichols:

Este es uno de los métodos más antiguos y populares para la sintonización de controladores PID. Se basa en la determinación de la ganancia crítica y el periodo crítico del sistema y luego ajusta los parámetros del controlador PID en función de estos valores. Este método se fundamenta en la observación empírica de la respuesta del sistema a una entrada escalón. El método implica la medición del tiempo de subida, el tiempo de establecimiento y la frecuencia natural del sistema. A partir de estas mediciones, se calculan las ganancias del controlador proporcional, integral y derivativo. En el método de Ziegler-Nichols, se realiza una prueba de respuesta escalón para determinar el periodo de oscilación y el tiempo de retardo del sistema. A partir de estos valores se calculan los parámetros del controlador. [3]

Este método es uno de los métodos de sintonía de controladores PID más conocidos y utilizados. La ventaja de este método es que es relativamente simple de implementar y no requiere conocimientos avanzados de teoría de control. Sin embargo, la desventaja es que puede producir una respuesta sobreamortiguada, lo que puede afectar la estabilidad del sistema.

Este método es adecuado para sistemas que no son muy críticos en términos de estabilidad, como los sistemas de calefacción y refrigeración. Las principales ventajas de emplear este método de sintonización son la facilidad de aplicación y que no requiere una comprensión profunda de la teoría del control. Se puede utilizar para ajustar los controladores PID de sistemas de la vida real. Sin embargo, los resultados pueden ser imprecisos debido a las variaciones en el sistema y las mediciones de respuesta. Puede no funcionar bien para sistemas no lineales o de respuesta no uniforme.

Ziegler y Nichols propusieron varias pautas para ajustar controladores PID basadas en la respuesta temporal del sistema a un cambio brusco en la entrada. El objetivo inicial de estas directrices era diseñar controladores cuando no se dispone de un modelo matemático de la planta o cuando el modelo matemático de la planta es tan complejo que obtenerlo resulta difícil. En tales casos, el enfoque analítico para el diseño del controlador PID no es factible.

Sin embargo, estas directrices también se aplican en sistemas con modelos matemáticos conocidos como una técnica alternativa o como un enfoque experimental inicial para el diseño. En estos casos, las directrices de Ziegler-Nichols ofrecen una estimación razonada de los parámetros, proporcionando un punto de partida para un diseño o ajuste posterior.

La determinación de los parámetros mediante las directrices de Ziegler-Nichols se realiza experimentalmente en la propia planta. Ziegler-Nichols propuso dos conjuntos de directrices. En ambos casos, el objetivo es lograr un máximo sobrepaso del 25%. [4]

Método 1:

En primer lugar, se debe obtener la respuesta del sistema ante un cambio brusco de entrada unitaria. Si la planta no contiene integradores ni polos dominantes complejos conjugados, la curva de respuesta puede tener forma de "S". A partir de esta curva, se pueden definir y calcular dos parámetros. Estos dos parámetros se determinan trazando una recta tangente en el punto de inflexión de la curva de respuesta:

El tiempo de retardo, L, se determina mediante la intersección de la tangente con el eje del tiempo.

La constante de tiempo, T, se determina mediante la intersección de la tangente con la recta y(t) = K.

La función de transferencia del sistema Y(s)/X(s) se puede aproximar utilizando un sistema de primer orden con retardo de transporte de la siguiente manera:

$$\frac{Y(s)}{X(s)} = \frac{K * e^{-Ls}}{Ts + 1}$$

En el primer método de Ziegler-Nichols se sugiere establecer los valores de Kp, Ti y Td según las expresiones de la Tabla 1.

Tabla 1. Ziegler-Nichols I

Tipo de controlador	Kp	Ti	Td
P	$\frac{T}{L}$	∞	0
PI	$\frac{0.9 * T}{L}$	$\frac{L}{0.3}$	0
PID	$\frac{1.2 * T}{L}$	2L	0.5L

Método 2:

En este enfoque, se siguen los siguientes pasos:

Comenzamos con los valores Ti = ∞ y Td = 0. Utilizando únicamente la acción proporcional, incrementamos Kp desde 0 hasta alcanzar un valor crítico, Kpcr,

donde la salida presenta oscilaciones sostenidas. Esto nos permite determinar la ganancia crítica Kpcr y el período crítico Pcr.

Establecemos los valores de Kp, Ti y Td según las expresiones indicadas en la Tabla 2.

Tabla 2. Ziegler Nichols II

Tipo de controlador	Kp	Ti	Td
P	$0.5 * Kp_{cr}$	∞	0
PI	$0.45 * Kp_{cr}$	$\frac{1}{1.2} * P_{cr}$	0
PID	$0.6 * Kp_{cr}$	$0.5 * P_{cr}$	$0.125 * P_{cr}$

El controlador PID obtenido mediante este segundo método se expresa como: $Gr(s) = Kp(1 + \frac{1}{Ti*s} + Td * s)$.

Existen sistemas en los que no es aplicable el método de Ziegler-Nichols. Por ejemplo, en un sistema de control con retroalimentación unitaria que posee un integrador, puede no ser aplicable el primer método, porque la respuesta no siga la forma de una curva en S. Si además el sistema en lazo cerrado no presenta oscilaciones sostenidas, el segundo método tampoco podría ser aplicado.

2.3.2. Método de Cohen-Coon:

Se basa en la estimación de dos constantes del sistema ($\theta \; y \; \tau$), que se pueden utilizar para calcular los parámetros del controlador PID. Este método es similar al método de sintonización Ziegler-Nichols, pero se basa en la respuesta del sistema a una entrada en rampa en lugar de una entrada escalón.

El método implica la medición del tiempo de subida y el tiempo de establecimiento y la determinación de una ecuación para las ganancias del controlador proporcional, integral y derivativo. Se utiliza una aproximación de primer y segundo orden para determinar los parámetros del controlador.

Este método es una mejora del método de Ziegler-Nichols y se utiliza para sistemas más críticos en términos de estabilidad. La ventaja de este método es que puede producir una respuesta menos sobreamortiguada y más estable que el método de Ziegler-Nichols. La desventaja es que puede ser más difícil de implementar y puede requerir conocimientos más avanzados de teoría de control. También, puede ser difícil de aplicar a sistemas con retardos grandes o no lineales y no siempre produce resultados óptimos. Este método es adecuado para sistemas que son críticos en términos de estabilidad, como los sistemas de control de procesos químicos. [5]

Tabla 3. Cohen-Coon

Tipo de controlador	Kp	Ti	Td
P	$K_c = \frac{1}{K}\frac{\tau}{\theta}(1 + \frac{\theta}{3\tau})$	-	-
PI	$K_c = \frac{1}{K}\frac{\tau}{\theta}(0.9 + \frac{\theta}{12\tau})$	$K_c = \theta\frac{30 + 3\frac{\theta}{\tau}}{9 + 20\frac{\theta}{\tau}}$	-
PID	$K_c = \frac{1}{K}\frac{\tau}{\theta}(\frac{4}{3} + \frac{\theta}{4\tau})$	$K_c = \theta\frac{32 + 6\frac{\theta}{\tau}}{13 + 8\frac{\theta}{\tau}}$	$K_c = \theta\frac{4}{11 + 2\frac{\theta}{\tau}}$

2.3.3. Método de Tyreus-Luyben:

Este método también se basa en la respuesta al escalón del sistema y utiliza una fórmula matemática para determinar los parámetros del controlador PID, como se refleja en la Tabla 4. Este método utiliza una técnica de ajuste iterativo para sintonizar los parámetros del controlador PID. Se requiere una prueba en la que se aplique una perturbación al sistema y se mida la respuesta.

Se lleva a cabo una evaluación exhaustiva de los parámetros del regulador, basándonos en la medida conocida como ganancia crítica K_{cu} y el período crítico τ_u. Este método se aplica fundamentalmente a plantas que poseen un integrador.

Tabla 4. Tyreus-Luyben

Tipo de controlador	Kp	Ti	Td
PI	$\frac{K_{cu}}{3.2}$	$\frac{\tau_u}{0.45}$	0
PID	$\frac{K_{cu}}{3.2}$	$\frac{\tau_u}{0.45}$	$\frac{\tau_u}{6.3}$

2.3.4. Método de IMC (Internal Model Control):

Este método utiliza un modelo interno del sistema para ajustar los parámetros del controlador PID. El modelo interno se utiliza para determinar los efectos de los cambios en los parámetros del controlador en el sistema. Se basa en la

identificación de un modelo interno del sistema y en la sintonización del controlador PID en función de ese modelo. Se busca que el controlador proporcione una respuesta similar a la del modelo interno.

El diagrama de bloques con modelo interno (IMC), representado en la Ilustración 5, responde a la idea intuitiva de que, formalmente, la realimentación debe manejar las diferencias entre simulación y medida. Sólo funciona con procesos estables (si son inestables, esa diferencia tiende a infinito).

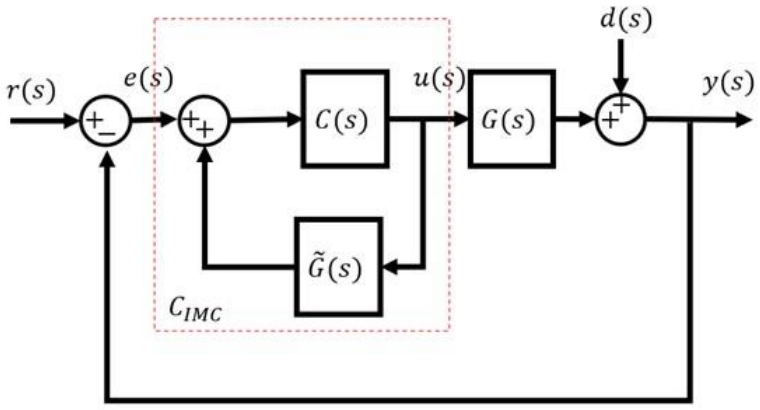

Ilustración 5. Diagrama bloques IMC

2.3.5. Método básico basado LDR:

Por otro lado, el enfoque basado en el diseño implica el uso de modelos matemáticos del sistema para diseñar el controlador.

El método de diseño de controladores PID básico basado en el lugar de las raíces, es el método más utilizado y consiste en modificar la ubicación de las raíces del sistema original mediante la incorporación de polos o ceros a través del controlador, con el propósito de cumplir con los requisitos estáticos y dinámicos del sistema.

Las raíces de la ecuación característica de un sistema establecen la estabilidad absoluta y relativa del mismo. Además, la respuesta momentánea de un sistema es influenciada por la ubicación de las raíces de su fórmula característica. Con el fin de realizar un análisis exhaustivo del sistema, resulta imprescindible conocer la ubicación de los polos en el plano complejo. Para obtener los polos en lazo cerrado de manera analítica, es necesario descomponer la fórmula característica, lo cual suele ser bastante complejo en la práctica. Además, si alguna variable del sistema en lazo abierto experimenta cambios después de calcular los polos en lazo cerrado, se requeriría volver a descomponer el sistema para llevar a cabo un nuevo análisis. [6]

El lugar de las raíces consiste en un conjunto de reglas directas, mediante las cuales es posible determinar la ubicación de las raíces de la ecuación característica

cuando uno de los parámetros de la función de transferencia del sistema en lazo abierto varía desde menos infinito hasta más infinito. Este método permite, por ende, conocer la posición de los polos de un sistema en lazo cerrado (ceros de la fórmula característica) a partir del conocimiento de las polos y ceros del sistema en lazo abierto. En la Ilustración 6, se muestra un ejemplo de un cuaderno implementado que analiza el LDR paso por paso y de forma manual. Se ahonda más sobre esta herramienta en el capítulo 3.2.

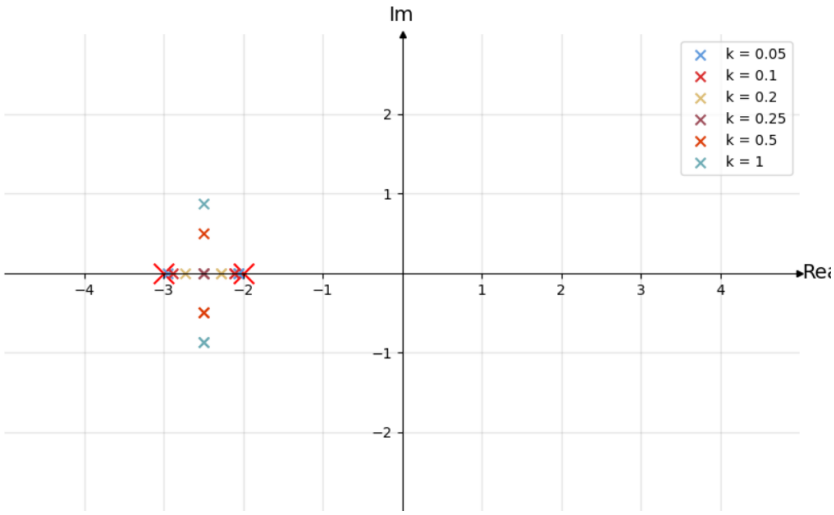

Ilustración 6. LDR en función valores k

El método de diseño basado en el lugar de las raíces se utiliza para seleccionar los valores de las constantes Kp, Ki y Kd en función de los requisitos de rendimiento y estabilidad del sistema. Este método utiliza el lugar de las raíces de la función de transferencia del sistema para ajustar las ganancias del controlador PID. Implica la selección de un punto deseado en el lugar de las raíces y la determinación de las ganancias del controlador necesarias para que el sistema tenga una respuesta deseada.

Concretamente, el método implica realizar ajustes apropiados en la dinámica de la planta con el fin de satisfacer los requisitos de diseño. Sin embargo, en la mayoría de los casos prácticos, la planta es estática y no se puede modificar. En tales situaciones, es necesario modificar parámetros distintos a los propios de la planta. Por lo tanto, el problema de control debe resolverse mejorando el comportamiento del sistema mediante la introducción de un regulador. Este regulador funciona como un filtro cuyas características se orientan a compensar los aspectos indeseables e inalterables de la planta.

En el enfoque de diseño, se coloca un regulador en serie con la función de transferencia inalterable de la planta, Gp(s), para lograr el comportamiento deseado. A continuación, el desafío principal radica en seleccionar los polos y los ceros apropiados para el regulador Gr(s), con el fin de alterar la ubicación geométrica de las raíces y así cumplir con las especificaciones requeridas.

Mediante el método del lugar de las raíces, se busca introducir un elemento de compensación en el bucle de control que modifique el comportamiento del sistema, de manera que sea aceptable tanto en régimen transitorio como en régimen permanente. El diseño de sistemas de control parte de la necesidad de cumplir con ciertas especificaciones tanto en el régimen transitorio como en el permanente. Si el sistema no satisface estas especificaciones, se incorporará el regulador Gr(s).

Para evaluar si el comportamiento de un sistema cumple con las especificaciones, es habitual analizar su respuesta ante una señal escalón unitario. Se ha comprobado experimentalmente que, si la respuesta de un sistema ante dicho escalón es aceptable, el sistema generalmente responderá bien ante cualquier tipo de entrada.

Como se ha mencionado, el lugar de las raíces permite observar la posición de los polos del sistema en lazo cerrado al variar el factor de ganancia K. Esto permite analizar el comportamiento del sistema en función de K. Por lo tanto, al tener unas especificaciones de diseño que deben cumplirse, se intentará ajustar el parámetro K de modo que la ubicación de los polos dominantes del sistema permita cumplir dichas especificaciones de manera satisfactoria.

El diseño de controladores PID mediante este método nos permite tener una comprensión clara de cómo el controlador afecta el comportamiento del sistema y adquirir las ganancias del controlador que optimizan la respuesta del sistema. Sin embargo, requiere una comprensión sólida del análisis del lugar de las raíces y puede ser difícil de aplicar a sistemas complejos con múltiples polos y ceros.

2.3.6. Método respuesta en frecuencia:

Existen varios métodos cuyo enfoque se basa en la respuesta del sistema en el dominio de la frecuencia. Estos métodos permiten seleccionar los valores de las constantes del controlador en función de los requisitos de rendimiento y estabilidad del sistema.

El método de diseño de controladores PID de respuesta en frecuencia o método de Bode se utiliza para ajustar los valores de las constantes Kp, Ki y Kd en función de la respuesta en frecuencia del sistema. Es uno de los métodos más utilizados para diseñar el controlador PID en el dominio de la frecuencia. Este método se basa en la representación del sistema y del controlador en un gráfico logarítmico de amplitud y fase en función de la frecuencia. Permite analizar la estabilidad y el rendimiento del sistema en el dominio de la frecuencia y ajustar los valores de las constantes Kp, Ki y Kd en función de la respuesta en frecuencia del sistema. Con este método, se puede ajustar la ganancia del controlador para conseguir una respuesta adecuada y satisfactoria en términos de estabilidad y rendimiento.

A diferencia del método del lugar de las raíces, no se determinan directamente los polos en la configuración en bucle cerrado. Además, las especificaciones en

términos de frecuencia ofrecen una estimación aproximada de las propiedades temporales del sistema, tales como el amortiguamiento, la estabilidad, la velocidad de respuesta y la precisión en estado estable.

El enfoque basado en el dominio de la frecuencia se aplica a sistemas cuyas características dinámicas se encuentran disponibles en forma de datos experimentales de respuesta en frecuencia. En algunos casos, resulta complicado obtener las ecuaciones que describen el comportamiento de componentes físicos, por lo que sus características dinámicas se determinan mediante pruebas experimentales de respuesta en frecuencia. Cuando se trabaja con ruido de alta frecuencia, el método de diseño en el dominio de la frecuencia resulta más conveniente que el enfoque temporal.[4]

El diseño en el dominio de la frecuencia se caracteriza por ser relativamente simple y directo. Las representaciones gráficas de la respuesta en frecuencia indican de manera clara cómo se debe modificar el sistema, aunque no sea posible realizar una predicción cuantitativa exacta de las propiedades transitorias de la respuesta del sistema controlado.

Por tanto, el diseño mediante este método, presenta como ventaja que es adecuado para sistemas que tienen una respuesta en frecuencia conocida y bien definida. También, proporciona una mejora en la precisión de la respuesta del sistema en comparación con algunos otros métodos. En cambio, requiere mediciones precisas de la respuesta en frecuencia del sistema y puede ser difícil de aplicar a sistemas con una respuesta en frecuencia no lineal o compleja.

Además de estos métodos, existen más métodos de diseño de controladores PID como el método de algoritmos genéticos y el método de sintonización LQR, que no requieren ajustar los parámetros del controlador en función de las respuestas del sistema real. Estos métodos permiten seleccionar los valores óptimos de las constantes del controlador en función de los requisitos de rendimiento y estabilidad del sistema de control.

2.3.7. Método algoritmos genéticos

El método de algoritmos genéticos es una técnica de optimización que se basa en la selección natural y la evolución biológica. Un algoritmo genético, conocido como AG en términos abreviados, constituye un software o una técnica de programación que se inspira en la emulación de organismos vivos y la simulación de la evolución biológica. Su objetivo principal radica en abordar problemas de optimización de manera estratégica. En líneas generales, los algoritmos genéticos (AG) se enmarcan dentro del campo de la inteligencia artificial, el cual se ocupa de resolver problemas mediante el empleo de programas informáticos que imitan el funcionamiento de la inteligencia natural.

Para aplicar un algoritmo genético para el diseño de controladores PID, se genera una población de posibles soluciones, que son combinadas y mutadas para generar nuevas soluciones. Estas soluciones son evaluadas utilizando una función objetivo

que mide el rendimiento del controlador. Las soluciones más prometedoras son seleccionadas para la siguiente generación, y el proceso continúa hasta que se alcanza una solución satisfactoria. Este enfoque es útil cuando el modelo matemático del sistema es desconocido o complejo, y se pueden explorar de manera eficiente grandes espacios de soluciones posibles. Puede optimizar las ganancias del controlador de manera efectiva para una respuesta del sistema mejorada. Es adecuado para sistemas no lineales y de alta complejidad. Sin embargo, puede ser computacionalmente costoso y requiere una comprensión sólida de la teoría del control. Puede no producir resultados precisos en situaciones en las que la respuesta del sistema es compleja o no se puede modelar fácilmente. [7]

2.3.8. Método LQR

Por último, el último método de diseño o sintonización que describiremos en el presente trabajo será el método de sintonización LQR (Regulador lineal cuadrático). Es un método de control utilizado en sistemas dinámicos para lograr un rendimiento perfecto y utiliza criterios cuadráticos para determinar la ley de control óptima.

Este método busca diseñar una ley de control lineal que minimice una función cuadrática, que representa la diferencia entre el comportamiento deseado del sistema y su estado real. Esta función típicamente incluye términos cuadráticos que ponderan tanto el error de seguimiento como el esfuerzo de control aplicado.

El objetivo principal del LQR es encontrar los valores óptimos de las ganancias del controlador que minimicen la función cuadrática. Esto se logra mediante la solución de las ecuaciones de Riccati, que proporcionan una solución matricial para las ganancias del controlador óptimo.

El regulador lineal cuadrático es ampliamente utilizado en diversas áreas de ingeniería y control de sistemas, ya que permite diseñar leyes de control robustas y eficientes. Se aplica en sistemas de control de procesos, robótica, control de vehículos y muchas otras aplicaciones donde se busca optimizar el rendimiento del sistema en función de criterios cuadráticos de rendimiento.

Lo mejor de este método es que proporciona una respuesta óptima del sistema en comparación con algunos otros métodos y es adecuado para sistemas de alta complejidad con un modelo matemático completo. Aunque, por otro lado, requiere un conocimiento profundo de la teoría del control y la modelización matemática del sistema y puede ser costoso y difícil de aplicar en la práctica.

2.4. Entornos para la implementación y diseño de controladores

En el ámbito del diseño de controladores PID, existen diferentes lenguajes, programas y herramientas que pueden utilizarse para llevar a cabo esta tarea. A continuación, compararé las ventajas y desventajas de utilizar los entornos tanto para el diseño de controladores PID como para el análisis, estudio y comparación de sistemas de control.

2.4.1. Matlab y Simulink

Matlab es un entorno de programación y desarrollo ampliamente utilizado en ingeniería y ciencias. Ofrece una amplia gama de herramientas y funciones para el diseño y análisis de sistemas de control. Matlab es un programa que nos permite programar directamente sin la necesidad de instalar un compilador externo ni instalar paquetes. Esto facilita el desarrollo de algoritmos y la implementación de controladores. Una de sus principales ventajas es que presenta una amplia variedad de funciones y librerías especializadas en el análisis y diseño de sistemas de control, lo que permite una implementación rápida y eficiente de controladores PID. También proporciona herramientas avanzadas de simulación y visualización, lo que facilita la comprensión y evaluación del comportamiento de los sistemas de control.

Pero lo más importante es que Matlab posee una toolbox llamada Simulink, que es una herramienta gráfica de Matlab que permite el modelado, simulación y análisis de sistemas dinámicos. Es ampliamente utilizado en el campo del control debido a su enfoque visual y su capacidad para simular y validar modelos de controladores PID en tiempo real. Simulink proporciona una interfaz gráfica intuitiva que permite el diseño y la simulación de sistemas de control mediante la conexión de bloques funcionales. Ofrece capacidades avanzadas de simulación y análisis, incluyendo la capacidad de realizar pruebas de hardware en bucle cerrado utilizando controladores reales.

Sin embargo, presenta dos grandes inconvenientes que han sido determinantes a la hora de seleccionar el entorno empleado para este proyecto. El primero de ellos es que Matlab es un software comercial que requiere una licencia, que implica un costo económico elevado para acceder a todas sus funcionalidades. Y el otro motivo es acerca de la complejidad de programar con este Software. Aunque el uso de Matlab no es especialmente difícil, sí puede requerir cierto tiempo para familiarizarse con su sintaxis y funciones.

Python, en comparación es mucho más fácil e intuitivo y como uno de los enfoques del trabajo es que sea en una herramienta que pueda servir para el estudio y análisis de sistemas para todo tipo de usuarios, se decidió descartar el uso de Matlab. Sin embargo, si es interesante y se ha empleado como segunda herramienta para verificar que los cálculos realizados sean correctos.

2.4.2. Python

Python es un lenguaje de programación de código abierto que está cada vez más extendido por su facilidad de uso. El análisis de sistemas de control y el diseño de controladores PID a través de Python, presenta numerosas ventajas por las cuales se ha decidido desarrollar el trabajo con este lenguaje de programación.

Primeramente, es un lenguaje de código abierto y gratuito, lo que lo hace accesible para cualquier persona. No hay restricciones de licencia y es posible utilizar numerosas bibliotecas especializadas en control y automatización como ControlPy y SciPy, además de otras como NumPy que sirven para optimizar cálculos con vectores y matrices multidimensionales y ofrecer una gran colección de funciones matemáticas. [8]

Otro punto fuerte de Python, es poder implementar programas y desarrollar herramientas siguiendo la estructura de Jupyter Notebook. Esto supone una ventaja en cuanto a optimizar y ordenar los programas, por lo que resulta de especial interés para proyectos de auto análisis como este. Ese punto se desarrolla más adelante en el capítulo 3.1.

A pesar de las ventajas mencionadas, también hay algunas consideraciones a tener en cuenta al utilizar Python:

- Rendimiento: En comparación con Matlab, Python puede ser menos eficiente en términos de rendimiento computacional, especialmente en aplicaciones que requieren cálculos intensivos en tiempo real.
- Dependencia de bibliotecas externas: Si bien Python cuenta con una amplia gama de bibliotecas y herramientas especializadas para el diseño de controladores PID y el análisis de sistemas de control, es posible que deba depender de bibliotecas externas para acceder a ciertas funcionalidades específicas. Esto puede requerir la instalación y gestión de dichas bibliotecas adicionales.
- Menor disponibilidad de funciones específicas para control: Aunque Python ofrece bibliotecas como SciPy y Control que contienen funciones para el diseño de controladores y el análisis de sistemas de control, su oferta puede ser menor en comparación con las herramientas y funciones especializadas proporcionadas por Matlab.

2.4.3. C/C++:

C y C++ son lenguajes de programación de propósito general ampliamente utilizados en el desarrollo de software y sistemas embebidos. Aunque no son específicos para el diseño de controladores PID, son utilizados en aplicaciones de

control en tiempo real y sistemas de bajo nivel. Algunas ventajas de utilizar C/C++ son:

Rendimiento: C/C++ se caracterizan por su eficiencia y capacidad de ejecución rápida, lo que los hace adecuados para aplicaciones con requisitos de tiempo real y alto rendimiento.

Control de hardware: C/C++ permite un mayor control sobre el hardware, lo que puede ser beneficioso en aplicaciones de control que requieren una interacción directa con dispositivos o sistemas físicos.

Sin embargo, algunas desventajas de C/C++ son:

Curva de aprendizaje: C/C++ es un lenguaje de programación de bajo nivel que requiere un mayor nivel de conocimiento y experiencia en comparación con Matlab o Python.

Mayor complejidad: El desarrollo en C/C++ implica una mayor complejidad y más líneas de código en comparación con entornos gráficos o lenguajes de alto nivel.

2.4.4. LabVIEW:

LabVIEW es un entorno de desarrollo de sistemas y control basado en gráficos, desarrollado por National Instruments. Se utiliza ampliamente en aplicaciones de control y adquisición de datos. Algunas ventajas de LabVIEW para el diseño de controladores PID son:

Programación gráfica: LabVIEW utiliza un enfoque gráfico de programación mediante la conexión de bloques funcionales, lo que facilita el diseño y la implementación de controladores PID.

Amplia variedad de módulos y herramientas: LabVIEW ofrece una amplia gama de módulos y herramientas especializadas en control y adquisición de datos, lo que permite una implementación rápida y eficiente de controladores PID.

Compatibilidad con hardware de adquisición de datos: LabVIEW se integra estrechamente con hardware de adquisición de datos, lo que facilita la interacción y el control de sistemas físicos en tiempo real.

Sin embargo, algunas desventajas de LabVIEW son:

Licencia y costos: LabVIEW es un software comercial que requiere una licencia y puede tener costos asociados.

Curva de aprendizaje: La programación gráfica utilizada en LabVIEW puede requerir cierto tiempo para familiarizarse, especialmente para aquellos sin experiencia previa en este entorno.

2.5. Aplicaciones controladores PID

Las aplicaciones de los controladores PID son amplias y se utilizan en diversos campos, como la industria, la robótica, la automoción y los sistemas de control de procesos. Algunos ejemplos notables son:

Control de temperatura: Los controladores PID se utilizan para mantener la temperatura de sistemas como hornos industriales, cámaras climáticas y sistemas de calefacción y refrigeración dentro de rangos específicos. La combinación de los términos proporcional, integral y derivativo permite un control preciso y estable de la temperatura.

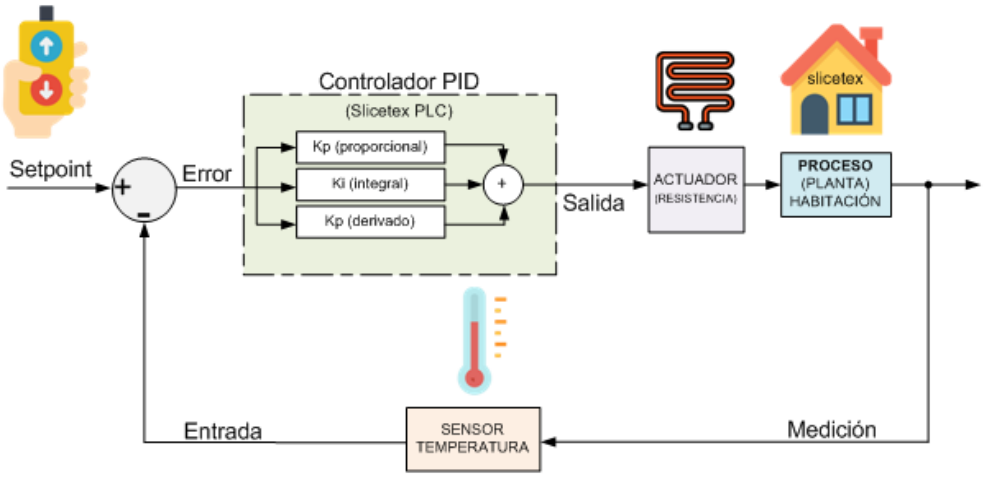

Ilustración 7. Ejemplo esquema control temperatura

Control de velocidad: Los controladores PID se emplean en motores eléctricos y sistemas de accionamiento para regular la velocidad en función de la carga y las condiciones variables. Esto es especialmente útil en aplicaciones donde se requiere un control preciso de la velocidad, como en máquinas herramienta, robots y vehículos autónomos.

Control de nivel: En sistemas de tanques, reactores químicos o plantas de tratamiento de agua, los controladores PID se utilizan para mantener un nivel constante. El controlador ajusta la entrada o salida de líquido según la medición del nivel, evitando desbordamientos o niveles insuficientes.

Control de posición: Los controladores PID se emplean en sistemas de posicionamiento, como brazos robóticos, sistemas de seguimiento solar o sistemas de control de vuelo de aeronaves. El controlador PID garantiza una respuesta precisa y estable para alcanzar la posición deseada.

2.5.1. Controladores PID en la robótica

Ahondando en las aplicaciones de controladores PID en el campo de la robótica, los controladores PID han demostrado ser una herramienta efectiva para mejorar el rendimiento y la precisión de los sistemas robóticos. A lo largo de este apartado,

se presentarán ejemplos detallados de cómo los controladores PID se utilizan en diversas aplicaciones robóticas, desde el control de posición hasta la navegación y la manipulación de objetos.

- Control de Posición Preciso

Una de las aplicaciones más comunes de los controladores PID en robótica es el control de posición preciso de manipuladores robóticos. Los controladores PID permiten que los robots se muevan con precisión a ubicaciones específicas en un espacio tridimensional. Al ajustar continuamente la señal de control enviada a los actuadores del robot en función del error de posición, el controlador PID logra minimizar dicho error y lograr una posición objetivo con alta precisión y estabilidad.[9]

- Control de Movimiento Suave

Otra aplicación importante de los controladores PID en robótica es el control de movimiento suave. Los controladores PID permiten que los robots se desplacen de manera fluida y sin oscilaciones indeseadas. Al ajustar los parámetros del controlador PID adecuadamente, se logra un seguimiento suave de trayectorias, lo que es esencial en aplicaciones como el ensamblaje de piezas o el movimiento de robots en entornos estrechos y delicados.

- Navegación Autónoma

La navegación autónoma es un campo en rápido crecimiento en robótica, y los controladores PID desempeñan un papel fundamental en esta área. Los robots móviles utilizan controladores PID para mantener una trayectoria deseada mientras evitan obstáculos. Al recibir información de sensores, como cámaras y sensores de proximidad, el controlador PID ajusta las velocidades de las ruedas del robot para seguir una ruta predefinida con alta precisión y seguridad.

- Control de Fuerza y Tensión

En aplicaciones donde se requiere un control preciso de fuerza y tensión, los controladores PID son ampliamente utilizados. Por ejemplo, en aplicaciones de ensamblaje y manipulación de objetos delicados, los controladores PID ajustan la fuerza ejercida por los actuadores para evitar daños. Esto se logra mediante la retroalimentación continua de sensores de fuerza y la adaptación de los parámetros del controlador PID para mantener la fuerza y tensión deseadas durante las interacciones robot-objeto.

3. Material y métodos

3.1. Herramientas

En este apartado, se describen las herramientas utilizadas en el desarrollo de la aplicación interactiva para el diseño de controladores PID basada en Python. Estas herramientas se seleccionaron por su capacidad para implementar y visualizar los resultados de los diferentes métodos de diseño y sintonización de controladores PID.

3.1.1. Jupyter Notebook:

Jupyter Notebook es una herramienta que permite crear y compartir documentos interactivos que contienen código, texto, visualizaciones y otros elementos. Es una herramienta de código abierto que admite varios lenguajes de programación, incluido Python. En este trabajo, se utilizó Jupyter Notebook para crear y ejecutar el código Python necesario para implementar los diferentes métodos de diseño y sintonización de controladores PID. Además, Jupyter Notebook permitió la visualización de los resultados de manera interactiva y detallada.

3.1.2. Google Colab:

Google Colab es una plataforma en línea que permite escribir y ejecutar código Python en la nube, sin necesidad de instalar ningún software en el equipo local. Es una herramienta gratuita que proporciona acceso a recursos de cómputo y almacenamiento en la nube de Google. En este trabajo, se utilizó Google Colab para ejecutar los códigos de Jupyter Notebook, lo que permitió un mayor rendimiento y capacidad de cómputo, así como la colaboración en tiempo real con otros usuarios.

3.1.3. Librerías:

En Python, las librerías son colecciones de funciones y métodos predefinidos que se pueden utilizar para realizar tareas específicas. En este trabajo, se utilizaron varias librerías para implementar los diferentes métodos de diseño y sintonización de controladores PID, incluyendo:

- NumPy: nos permite realizar cálculos numéricos con vectores y matrices multidimensionales y nos ofrece una gran colección de funciones matemáticas

- SciPy: se emplea para realizar cálculos científicos con Python. Posee numerosas funciones aplicables al control de sistemas.
- Sympy: sirve para realizar operaciones de un sistema en algebra computacional.
- Matplotlib: una librería para crear gráficos y visualizaciones con Python.
- Control: una librería específica para el diseño y análisis de sistemas de control con Python.
- Time: proporciona un conjunto de funciones para trabajar con fechas y/o horas.
- Random: permite generar números aleatorios.
- Math: es una biblioteca que incluye numerosas funciones matemáticas como la raíz, el logaritmo y el límite.
- Ipywidgets: proporciona accesorios que te permiten interaccionar con los datos de forma gráfica y en tiempo real, por ejemplo, los sliders. [10]

3.2. Funciones

3.2.1. Llamada de funciones entre cuadernos

Consideraciones previas:

Se debe trabajar en una Unidad Compartida de drive o en tu propia Unidad y no en una carpeta compartida.

Se crea y se guarda el cuaderno que contiene la función o funciones que deseamos llamar, por ejemplo, como la que se muestra en la Ilustración 8.

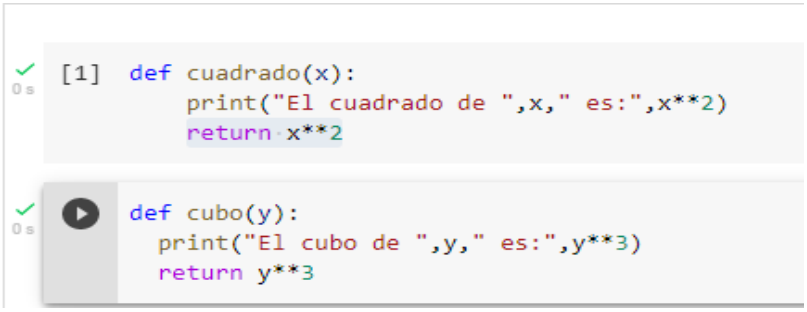

Ilustración 8. Ejemplo definir funciones

Un detalle importante es que el cuaderno no debe tener ningún pip install, ninguna llamada a otro cuaderno y ni ninguna salida, ya que se podrían ejecutar directamente al importar el cuaderno. Por tanto, conviene crear una copia del cuaderno borrando o comentando las líneas que no interesan antes de guardar este archivo.

Se guardan los cuadernos a los que vamos a llamar en formato .py dentro de la Unidad (Ilustración 9).

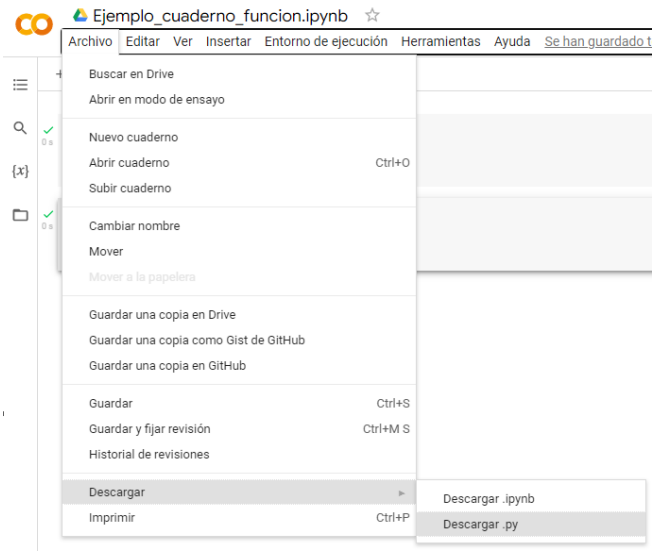

Ilustración 9. Descargar .py

Archivo → Descargar → Descargar .py

Podría ocurrir que el navegador detecte el archivo .py como un posible archivo dañino, sin embargo, no pasa nada. Se aceptan los permisos y se arrastra desde la carpeta de descargas del navegador a la Unidad Drive.

Se puede apreciar un ejemplo completo de llamada de función en la Ilustración 10.

```
from google.colab import files
```

+ Código + Texto

```
[9] from google.colab import drive

[15] drive.mount("/content/drive")

     Mounted at /content/drive

[18] !cp /content/drive/Shareddrives/Python_UNIDIGITAL_2022_2023/ejemplo_cuaderno_funcion.py /content

[19] import ejemplo_cuaderno_funcion

[20] ejemplo_cuaderno_funcion.cuadrado(5)

     El cuadrado de  5  es: 25
     25

     ejemplo_cuaderno_funcion.cubo(2)

     El cubo de  2  es: 8
     8
```

Ilustración 10. Ejemplo completo importar cuadernos

Se ejecutan las siguientes líneas:

from google.colab import files

from google.colab import drive

drive.mount("/content/drive")

!cp ruta_del_archivo /content →Ojo no se pueden poner espacios en las carpetas de la ruta del archivo.

Hay que instalar las librerías que requiera el cuaderno al que llamamos → pip install librería

import nombre_cuaderno_llamado →Ojo al descargar el cuaderno .py se guarda con el mismo nombre, pero todo en minúsculas, así que deberemos llamarle con éste nuevo nombre.

nombre_cuaderno_llamado.nombre_funcion(inputs)

Un atajo para hallar la ruta del archivo es: buscar la carpeta desde google colab, hacer click derecho sobre el cuaderno .py y seleccionar "Copiar ruta" (Ilustración 11).

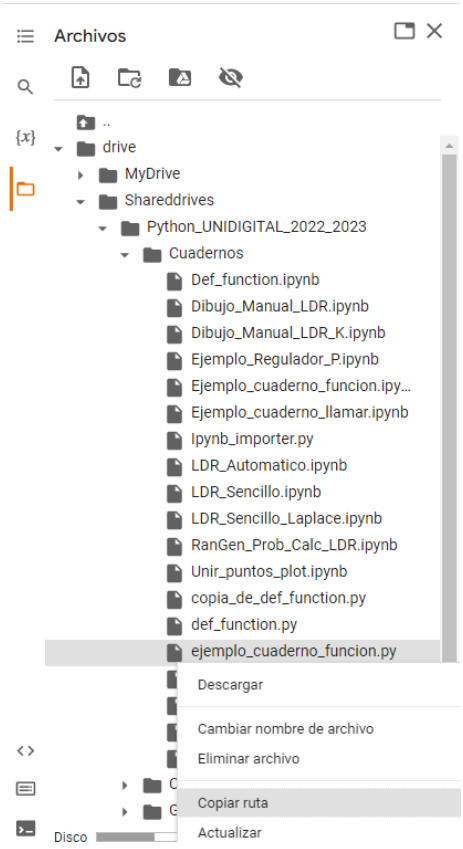

Ilustración 11. Atajo copiar Ruta

Problemas no actualiza drive:

Hay un problema con la actualización de cuadernos y en la llamada de estos.

Si después de montar el drive, modificamos algún archivo.py, no lo actualizará volviendo a ejecutar:

drive.mount("/content/drive",force_remount=True)

!cp /content/drive/Shareddrives/Python_UNIDIGITAL_2022_2023/Cuadernos/llamar_ldr_laplace.py /content

Hay que desmontar el entorno para actualizar los cambios que se hayan hecho en los cuadernos y volver a montarlo.

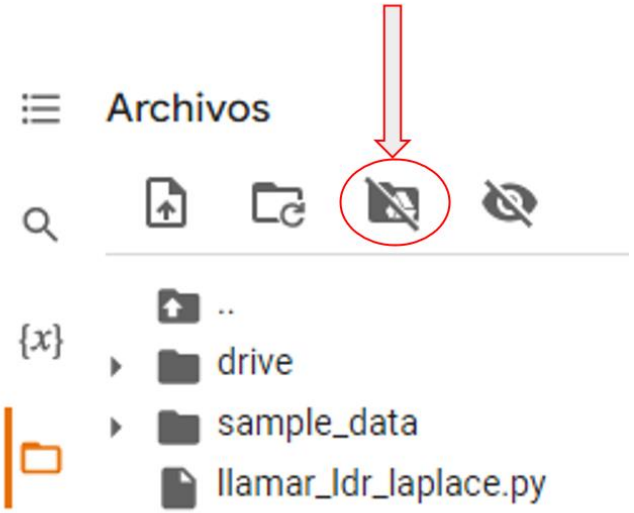

Ilustración 12. Desmontar

Una vez desmontado, se vuelve a montar con el mismo botón de la Ilustración 12. y se asigna nuevo entorno (Ilustración 13)

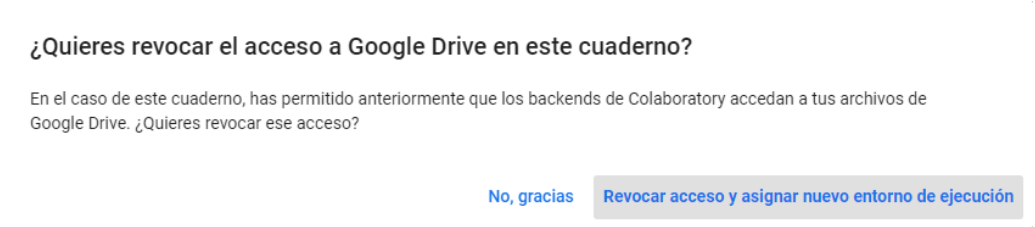

Ilustración 13. Revocar

3.2.2. Respuesta de sistemas

ver_respuesta_escalón:

Inputs: (pol_num,pol_den)

Es una función que muestra la gráfica de la respuesta del sistema independientemente del orden ante una entrada escalón, dándole como entradas 2 vectores que representan el índice de los polinomios del numerador y denominador de la función de transferencia del sistema (Ilustración 14).

```
pol_num = [1]
pol_den = [1, 6]
ver_respuesta_escalón(pol_num,pol_den)
```

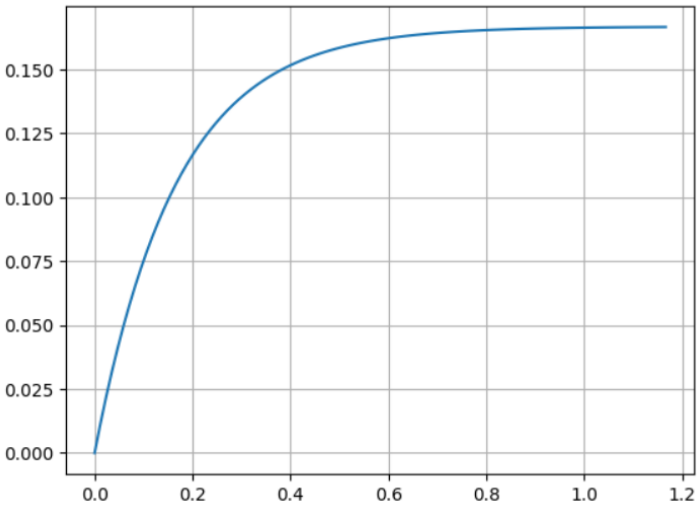

Ilustración 14. ver_respuesta_escalon

Respuesta_temporal:

Inputs: (pol_num,pol_den)

Output: (t,y)

Esta función devuelve como salida los valores que toma la función para cada instante de tiempo (Ilustración 15).

```
[5]  pol_num = [1]
     pol_den = [1, 6]
     [t,y]=Respuesta_temporal(pol_num,pol_den)

[6]  t

     array([0.000e+00, 1.000e-03, 2.000e-03, ..., 1.998e+00, 1.999e+00,
            2.000e+00])

     y

     array([0.        , 0.00099701, 0.00198805, ..., 0.16666563, 0.16666564,
            0.16666564])
```

Ilustración 15. respuesta_temporal

Respuesta_temporal_tmanual:

Inputs: (pol_num,pol_den,tmanual)

Output: (t,y)

Es igual que la función anterior, pero requiere introducir como tercer valor de entrada, el valor máximo del instante de tiempo de la señal de respuesta (Ilustración 16).

```
[22] pol_num = [1]
     pol_den = [1, 6]
     [t,y]=Respuesta_temporal_tmanual(pol_num,pol_den,4)
```

```
[23] t
```

```
array([0.000e+00, 1.000e-03, 2.000e-03, ..., 3.997e+00, 3.998e+00,
       3.999e+00])
```

```
y
```

```
array([0.        , 0.00099701, 0.00198805, ..., 0.16666667, 0.16666667,
       0.16666667])
```

Ilustración 16. Respuesta_temporal_tmanual

Respuesta_temporal_2:

Inputs: (M)

Output: (t,y)

Esta función al igual que las anteriores, devuelve como salida los valores que toma la función para cada instante de tiempo, sin embargo, tiene como entrada la función de transferencia en formato tf de la librería control. Se pueden comprobar en las Ilustraciones 17 y 18.

```
M
```

$$\frac{3}{s+4}$$

```
[t,y]=Respuesta_temporal_2(M)
```

```
t
```

```
array([0.        , 0.01744383, 0.03488765, 0.05233148, 0.06977531,
       0.08721913, 0.10466296, 0.12210679, 0.13955061, 0.15699444,
       0.17443826, 0.19188209, 0.20932592, 0.22676974, 0.24421357,
       0.2616574 , 0.27910122, 0.29654505, 0.31398888, 0.3314327 ,
       0.34887653, 0.36632036, 0.38376418, 0.40120801, 0.41865184,
       0.43609566, 0.45353949, 0.47098331, 0.48842714, 0.50587097,
       0.52331479, 0.54075862, 0.55820245, 0.57564627, 0.5930901 ,
       0.61053393, 0.62797775, 0.64542158, 0.66286541, 0.68030923,
       0.69775306, 0.71519688, 0.73264071, 0.75008454, 0.76752836,
       0.78497219, 0.80241602, 0.81985984, 0.83730367, 0.8547475 ,
       0.87219132, 0.88963515, 0.90707898, 0.9245228 , 0.94196663,
       0.95941046, 0.97685428, 0.99429811, 1.01174193, 1.02918576,
       1.04662959, 1.06407341, 1.08151724, 1.09896107, 1.11640489,
       1.13384872, 1.15129255, 1.16873637, 1.1861802 , 1.20362403,
       1.22106785, 1.23851168, 1.25595551, 1.27339933, 1.29084316,
       1.30828698, 1.32573081, 1.34317464, 1.36061846, 1.37806229,
       1.39550612, 1.41294994, 1.43039377, 1.4478376 , 1.46528142,
       1.48272525, 1.50016908, 1.5176129 , 1.53505673, 1.55250056,
       1.56994438, 1.58738821, 1.60483203, 1.62227586, 1.63971969,
       1.65716351, 1.67460734, 1.69205117, 1.70949499, 1.72693882])
```

Ilustración 17. Respuesta_temporal_2

```
y
```

```
array([0.        , 0.05054749, 0.09768825, 0.14165188, 0.1826525 ,
       0.22088983, 0.25655008, 0.28980695, 0.32082243, 0.34974756,
       0.37672323, 0.40188084, 0.4253429 , 0.44722371, 0.46762981,
       0.48666062, 0.50440881, 0.52096084, 0.53639731, 0.55079342,
       0.56421927, 0.57674027, 0.5884174 , 0.59930752, 0.60946369,
       0.61893537, 0.62776869, 0.63600667, 0.64368944, 0.65085441,
       0.65753649, 0.66376823, 0.66957996, 0.675     , 0.68005475,
       0.68476882, 0.68916519, 0.69326525, 0.69708898, 0.70065501,
       0.7039807 , 0.70708224, 0.70997476, 0.71267232, 0.71518808,
       0.71753429, 0.71972237, 0.72176298, 0.72366606, 0.72544088,
       0.72709608, 0.72863973, 0.73007934, 0.73142193, 0.73267403,
       0.73384174, 0.73493075, 0.73594637, 0.73689354, 0.73777687,
       0.73860067, 0.73936894, 0.74008544, 0.74075365, 0.74137682,
       0.741958  , 0.7425    , 0.74300547, 0.74347688, 0.74391652,
       0.74432653, 0.7447089 , 0.7450655 , 0.74539807, 0.74570822,
       0.74599748, 0.74626723, 0.74651881, 0.74675343, 0.74697224,
       0.7471763 , 0.74736661, 0.74754409, 0.74770961, 0.74786397,
       0.74800793, 0.74814219, 0.7482674 , 0.74838417, 0.74849308,
       0.74859464, 0.74868935, 0.74877769, 0.74886007, 0.74893689,
       0.74900854, 0.74907536, 0.74913768, 0.7491958 , 0.74925   ])
```

Ilustración 18. Respuesta_temporal_2 (y)

tipo_respuesta_2do_orden:

Inputs: (polos)

Outputs: (tipo)

Función que te devuelve el tipo de sistema de segundo orden en función del vector de polos de entrada que tenga el sistema. Sistema de 2do orden inestable → tipo=1, Sistema de 2do orden subamortiguado → tipo=2, Sistema de 2do orden críticamente amortiguado → tipo=3, Sistema de 2do orden sobreamortiguado → tipo=4

En la Ilustración 19, se puede ver un ejemplo de salida de esta función.

```
tipo_respuesta_2do_orden(polos_M)
```

```
Sistema de 2do orden críticamente amortiguado
3
```

Ilustración 19. tipo_respuesta_2do_orden

pintar_respuesta_1er_orden:

Inputs: (ax,t,y)

Función que representa la respuesta de un sistema de 1er orden, dando como entradas la variable imagen donde dibujar y los vectores de t e y con los distintos valores de la gráfica (Ilustración 20).

```
fig,ax = plt.subplots(figsize=(5, 5))
pintar_respuesta_1er_orden(ax,t,y)
```

Ilustración 20. pintar_respuesta_1er_orden

pintar_respuesta_2do_orden_subamortiguado:

Inputs: (ax,t,y,K,ts,tp,tr,Mp)

Función que representa la respuesta de un sistema de 2do orden de tipo subamortiguado, dando como entradas la variable imagen donde dibujar y los vectores de t e y con los distintos valores de la gráfica y los parámetros que caracterizan este sistema, K, ts, tp, tr y Mp (Ilustración 21).

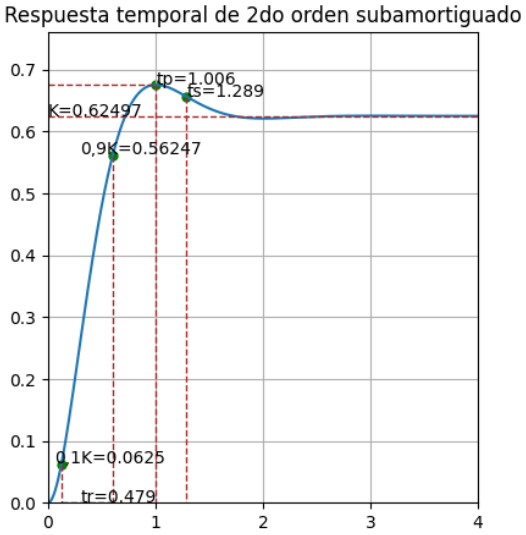

Ilustración 21. pintar_respuesta_2do_orden_subamortiguado

pintar_respuesta_2do_orden_subamortiguado_simplificado:

Inputs: (ax,t,y,K,ts,tp,tr,Mp)

Igual que la anterior función, pero con menos parámetros que dibujar en la gráfica. (Ilustración 22).

```
fig = plt.figure(figsize = (10,5))
ax1 = fig.add_subplot(1,2,1)
pintar_respuesta_2do_orden_subamortiguado_simplificado(ax1,t,y,K,ts,tp,tr,Mp)
```

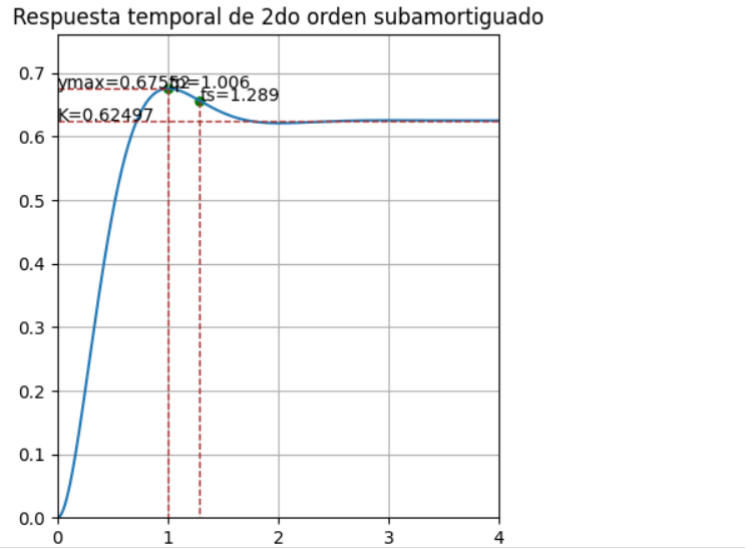

Ilustración 22. pintar_respuesta_2do_orden_subamortiguado_simplificado

pintar_respuesta_2do_orden_crit_amort:

Inputs: (ax,t,y)

```
fig,ax = plt.subplots(figsize=(5, 5))
pintar_respuesta_2do_orden_crit_amort(ax,t,y)
```

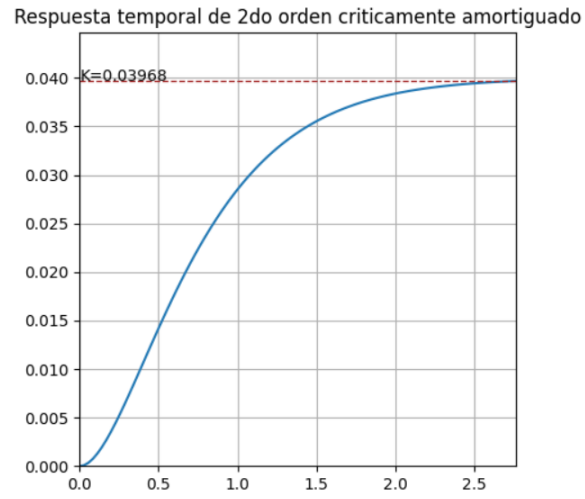

Ilustración 23. pintar_respuesta_2do_orden_crit_amort

pintar_respuesta_2do_orden_sobreamortiguado:

Inputs: (ax,t,y)

En la Ilustración 24, se aprecia la gráfica que genera esta función.

```
fig,ax = plt.subplots(figsize=(5, 5))
pintar_respuesta_2do_orden_sobreamortiguado(ax,t,y)
```

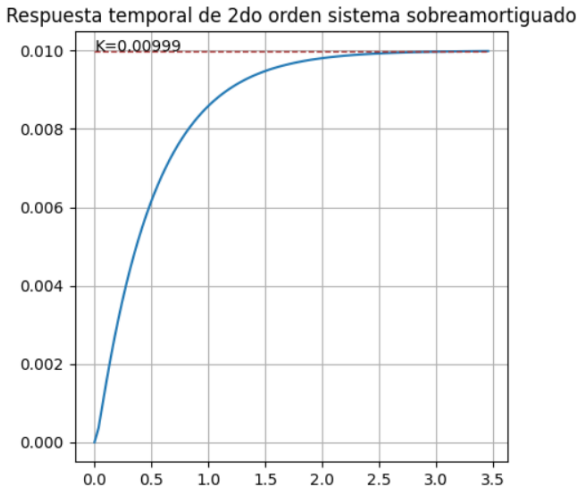

Ilustración 24. pintar_respuesta_2do_orden_sobreamortiguado

3.2.3. Representación en el plano complejo

Resuelve_TF_polinomio:

Inputs: (num,den)

Outputs: TF,ceros_TF,polos_TF

Resuelve_TF_polinomio_2:

Inputs:(num,den)

Outputs: TF

Dado los valores del polinomio numerador y denominador de una función de transferencia almacena la función en el formato tf de la librería de control y devuelve el vector de ceros y el vector de polos.

Resuelve_M_1:

Inputs: (G,H,k)

Outputs: M,ceros_M,polos_M

Resuelve_M_2:

Inputs: (Gp,Gr,H)

Outputs: M,ceros_M,polos_M

Dados la planta, el controlador y el sensor calcula y devuelve la función de transferencia del sistema, así como sus polos y ceros.

Devuelve_polinomio_num_den :

Inputs: (tf)

Outputs: ec1, ec2

Devuelve_polinomio_num_denç:

Inputs: (num,den)

Outputs: ec1, ec2

Dado una función con formato control.tf devuelve 2 polinomios correspondientes al numerador y denominador de la función de transferencia dada, para poder operar con ella.

Resuelve_TF_polos_y_ceros:

Inputs:(ceros_TF,polos_TF)

Output: TF

Dados el vector de polos y ceros de una función de transferencia devuelve la función de transferencia con formato control.tf

Serie:

Inputs: (sys1,sys2):

Output: lti(num,den)

Calcula la función de transferencia dados dos sistemas que se encuentran en serie.

Feedback:

Inputs: (sys1,sys2)

Output: lti(num,den)

Calcula la función de transferencia dados dos sistemas que se encuentran en paralelo.

definir_ejes:

Inputs:(G,H,lista_k)

Outputs: xmax,ymax

Dan los valores máximos que debe ocupar nuestra gráfica para que represente los polos y ceros del sistema dado de manera óptima.

pintar_ejes:

Inputs: (xmax,ymax,ax)

Pinta los ejes del sistema dando sus coordenadas máximas y en lugar de ploteo.

Pintar_G :

Inputs: (ceros_G,polos_G,ax)

Pinta los polos y ceros de G dándole el vector de los mismos.

Pintar_polos_k:

Inputs: (polos_M,k,color_polos,ax)

Pinta los polos de la función de transferencia del sistema M, dándole el vector de los mismos, el vector de valores de k, el color de los polos y la imagen donde dibujar (Ilustración 25).

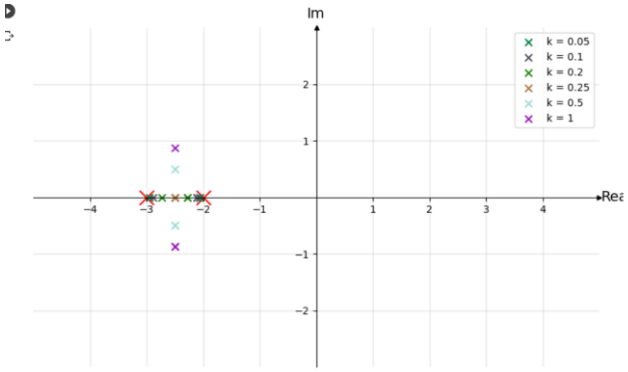

Ilustración 25. Funciones Puntar_G, pintar_polos y pintar_ejes

cambiar_color():

Output: color

Devuelve un color aleatorio en hexadecimal

cambiar_color_2():

Output: color

Igual que la función anterior, pero con colores más oscuros para distinguir mejor ciertas tonalidades; a costa de tener menor variedad en las tonalidades

LDR_Automático:

Inputs: (ax,ceros_G,polos_G,xmax,ymax)

Outputs: rlist, klist

Representa el LDR de G de forma automática (Ilustración 26) [11].

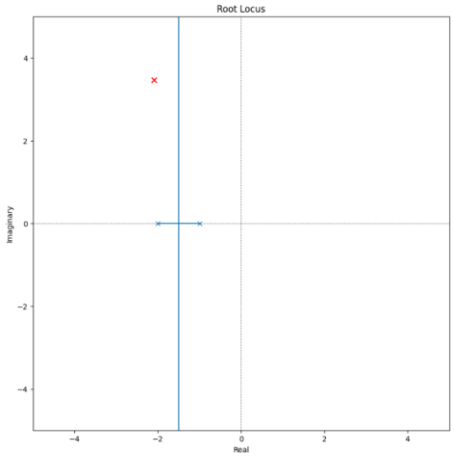

Ilustración 26. LDR Automático

3.2.4. Parámetros

Param:

Inputs: (t,y)

Outputs: Mp, vf, tp, tl, ts

Devuelve la sobreobscilación, el valor final, el tiempo de pico, el tiempo de subida y el tiempo de establecimiento.

calcula_theta:

Inputs: (Mp)

Output: theta

Despeja theta de la ecuación que define porcentaje de sobreoscilación

calcula_sigma:

Inputs: (ts)

Output: sigma

Calcula sigma para el tiempo de establecimiento dado

calcula_coef_amortiguamiento:

Inputs: (Mp)

Output: coef_amort

Calcula el coeficiente de amortiguamiento a través del porcentaje de sobreoscilación introduccido

calcula_wn:

Inputs: (sigma,coef_amort)

Output: wn

Con el coeficiente de amortiguamiento y el ángulo sigma, calcula wn

calcula_angulo_polo_o_cero:

Inputs: (Pd,polo_o_cero_real)

Output: angulo

Devuelve el ángulo del polo o cero dado, dicho polo o cero y el polo dominante

calculo_theta_d:

Inputs: (theta_1)

Output: theta_d

Calcula el ángulo del polo del controlador conocidos el resto de ángulos

calcula_polo_d:

Inputs: (theta_d,Pd)

Output: d_real

Calcula el polo del controlador conocido el ángulo de este sobre el polo dominante

3.3. Implementación métodos diseño PID

3.3.1. Método Manual

El primer método que se ha diseñado, a modo introductorio, es el denominado método manual. Este método requiere pasar como entradas la planta y el sensor del sistema, en forma de función scipy.tf. Gracias a la librería ipywidgets, en plena ejecución, se pueden ir variando los valores de kp, kd y ki a través de sliders. Cada vez que se varía el sistema mediante los parámetros del controlador, se reajustan los valores ep, tr, ts, Mp, tp que recogen las características propias del sistema y se proporciona la nueva grafica del sistema controlado.

```
def
metodo_manual(planta,H,kp_max=20,ki_max=20,kd_max=20,delta_P=0.01,delt
a_I=0.01,delta_D=0.01):

  @interact(kp=(ipywidgets.FloatSlider(min=0, max=kp_max, step=delta_P,
value=1)),ki=(ipywidgets.FloatSlider(min=0, max=ki_max, step=delta_I,
value=0)),kd=(ipywidgets.FloatSlider(min=0, max=kd_max, step=delta_D,
value=0)))
```

```
def CONTROLAR(kp,ki,kd):
  ts,ys=planta.step()
  cont=funciones.lti([kd,kp,ki],[1,0])
  ser=funciones.serie(cont,planta)
  fed=funciones.feedback(ser,H)
  #[sys_num,sys_den]=feedback(ser,H)
  #fed = funcion de transferencia del sistema
  tc,yc=fed.step(T=ts)
  funciones.plt.plot(tc,yc,label='Sistema controlado')
  funciones.plt.title('PID')
  funciones.plt.xlabel('T(s)')
  funciones.plt.ylabel('Amplitud')
  funciones.plt.grid()
  funciones.plt.show
  print('\n Parametros del sistema controlado')
  Mp,vf,tp,tr,tss=funciones.param(tc,yc)
  ep=1/(1+kp)
  print(f'ep={ep}')
  print(f'tr={tr}')
  print(f'ts={tss}')
  print(f'Mp={Mp}')
  print(f'tp={tp}')
  num=fed.num
  den=fed.den
  sys=funciones.ct.TransferFunction(num,den)
  s=funciones.ct.tf('s')
  print('M=',sys)
```

3.3.2. Método fuerza bruta

El método de fuerza bruta consiste en una tabla donde se evalúan todos los casos posibles de valores del controlador, visualizando como afecta esto a las especificaciones del sistema ep, tr, ts, Mp y tp. Requiere pasar como entradas los vectores de kp, kd y ki en formato np.arange y la planta y el sensor del sistema, en forma de función scipy.tf.

```
def metodo_fuerza_bruta_v2(vector_kp,vector_kd,vector_ki,planta,H):
  kp=0
  kd=0
  ki=0
  n=0
  kpp=np.empty(shape=110000)
  kdd=np.empty(shape=110000)
  kii=np.empty(shape=110000)
  tr=np.empty(shape=110000)
  tss=np.empty(shape=110000)
```

```
 Mp=np.empty(shape=110000)
 tp=np.empty(shape=110000)
 ep=np.empty(shape=110000)
 vf=np.empty(shape=110000)
 #a=['kd','kp','ki','tr','ts','Mp','tp','ep']
 #print(a)
 for kd in vector_kd:
   for  kp in vector_kp:
     for ki in vector_ki:
       cont=lti([kd,kp,ki],[1,0])
       ser=funciones.serie(cont,planta)
       fed=funciones.feedback(ser,H)
       tc,yc=fed.step()
       num=fed.num
       den=fed.den
       sys=ct.TransferFunction(num,den)
       s=ct.tf('s')
       Mp[n],vf[n],tp[n],tr[n],tss[n]=funciones.param(tc,yc)
       ep[n]=round(1/(1+kp),2)
       kdd[n]=round(kd,2)
       kpp[n]=round(kp,2)
       kii[n]=round(ki,2)
       #matriz=[[round(kd,2),round(kp,2),round(ki,2),round(tr[n],2),round(tss[n],2
),round(Mp[n],2),round(tp[n],2),round(ep,2)]]
       #print(matriz)
       #print(f'{n}. Para kd = {round(kd,2)}, kp = {round(kp,2)} y ki =
{round(ki,2)}, tr = {tr[n]}, ts = {tss[n]}, Mp = {Mp[n]}, tp = {tp[n]}')
       n=n+1
 #borrar huecos vacios de la matriz
 kp_=np.empty(shape=n)
 kd_=np.empty(shape=n)
 ki_=np.empty(shape=n)
 tr_=np.empty(shape=n)
 tss_=np.empty(shape=n)
 Mp_=np.empty(shape=n)
 tp_=np.empty(shape=n)
 ep_=np.empty(shape=n)
 vf_=np.empty(shape=n)
 for i in range(n):
   kp_[i]=kpp[i]
   kd_[i]=kdd[i]
   ki_[i]=kii[i]
   tr_[i]=tr[i]
   tss_[i]=tss[i]
   Mp_[i]=Mp[i]
   tp_[i]=tp[i]
   ep_[i]=ep[i]
   #if (kpp[i]==0 and kdd[i]==0 and kii[i]==0 and tr[i]==0 and tss[i]==0 and
Mp[i]==0 and tp[i]==0 and ep[i]==0):break
```

```
kd1=list(kd_)
kp1=list(kp_)
ki1=list(ki_)
tr1=list(tr_)
ts1=list(tss_)
Mp1=list(Mp_)
tp1=list(tp_)
ep1=list(ep_)
return kd1,kp1,ki1,tr1,ts1,Mp1,tp1,ep1
```

3.3.3. Método del Lugar de las Raíces

También se ha implementado el método básico y el más utilizado para el diseño de controladores PID que es el basado en el LDR. En función del sistema y las especificaciones requeridas, el hecho de diseñar un regulador de manera que el sistema presente las características deseadas puede ser una tarea que conlleve más o menos pasos. Las tres formas de hacer que un sistema cumpla los requisitos por medio de un regulador es mediante la compensación PD, la compensación PI y la compensación PID. Como norma general un sistema siempre se tratará de resolver con un compensador PD, luego con un compensador PI y por último con un PID (combinación de ambos anteriores).

1. Compensación PD:

La idea fundamental radica en la incorporación de un controlador de manera que la función de transferencia resultante, en la configuración de lazo cerrado del sistema, exhiba un par de polos complejos conjugados en el punto de operación deseado.

Con las especificaciones, se deben determinar los valores de ξ y w_n para calcular el punto de operación deseado. Para este propósito, se puede utilizar la expresión: $P_d = -\xi w_n \pm \mathrm{j} w_n \sqrt{1-\xi^2}$

Se debe trazar el diagrama del lugar de las raíces del sistema sin el controlador y verificar que no atraviese dicho punto. En caso contrario, bastaría con ajustar la ganancia.

En caso necesario, se procederá a introducir un controlador PD, calculando la ubicación del polo y del cero, de tal manera que el lugar de las raíces pase por los puntos especificados.

A continuación, se calculará la ganancia total del sistema con el controlador, a fin de que funcione en el punto deseado.

Finalmente, se determinará la constante de error del sistema con el controlador. Si este valor no es aceptable, se volverá a variar la posición del par polo-cero del controlador.

2. Compensación PI:

Estos controladores se utilizan principalmente para reducir o eliminar el error en estado estacionario sin afectar significativamente la respuesta transitoria. El objetivo de este controlador es mantener en las proximidades de los polos dominantes el mismo lugar de las raíces, aumentando al mismo tiempo la ganancia en lazo abierto para reducir el error.

Los pasos a seguir para calcular un controlador de tipo PI o un compensador de retardo de fase se pueden resumir de la siguiente manera:

Primeramente, se muestra la gráfica del lugar de las raíces del sistema sin controlador.

Luego, conviene calcular los puntos de operación en función de las especificaciones y, para esa ubicación, determinar la ganancia en lazo abierto del sistema.

Posteriormente, se calcula la constante de error del sistema y se compara la constante de error del sistema con los valores requeridos en las especificaciones y determinar el valor de β.

Una vez calculado β, se determina la ubicación del par polo-cero del controlador. Al igual que con el controlador PD, existen varios métodos para ubicar este par polo-cero. En la práctica, se suelen utilizar tres criterios:

- Se coloca el par polo-cero del controlador muy cerca del origen en comparación con los polos y ceros del sistema original en lazo abierto, ubicando el cero en el eje real negativo a una distancia del origen igual (aproximadamente) a 0,1d, donde d es la distancia desde el origen hasta el primer polo o cero del sistema en lazo abierto ubicado en dicho eje, excluyendo el origen.
- Se coloca el cero a una distancia igual a la sexta parte de la parte real de los polos dominantes Pd.
- Se coloca el par polo-cero cerca del origen de manera que la diferencia entre el ángulo del polo y el ángulo del cero, al unirlos con el polo dominante, sea inferior a cinco grados.

Una vez que se ha determinado el par polo-cero del regulador, se representa el lugar de las raíces del sistema con el controlador.

Para concluir, se verifica que la respuesta transitoria siga siendo aceptable con el nuevo controlador.

3. Compensación PID:

El controlador PID emplea las estrategias de control proporcionadas por un controlador PD para mejorar la respuesta durante la fase transitoria y un controlador PI para mejorar la respuesta en estado estacionario. En la práctica, es

posible obtener un controlador PID al combinar en cascada un controlador PD y otro PI. Sin embargo, por motivos económicos y de implementación física, ambos controladores se integran en un único componente de tipo PID, que permite ajustar ambas estrategias de control. [14]

A continuación, se detallan los pasos a seguir para dimensionar un controlador PID:

En primer lugar, se determinan los valores de ξ, wn y la constante de error (Kp, Kv o Ka según el tipo de sistema) requeridos para el sistema compensado, de acuerdo con las especificaciones.

Se dimensiona la estrategia PD para que la respuesta transitoria cumpla con las especificaciones (ξ, wn).

Si la constante de error, por ejemplo, Kv1, cumple la condición Kv1 ≥ Kv (especificada), solo se necesitará un controlador PD. En caso contrario, si β1Kv1 ≥ Kv, siendo β1 el valor obtenido en el paso anterior, se dimensionará la estrategia PI con β = β1. Si β1Kv1 < Kv, se elegirá un nuevo valor β = β2, de manera que β2Kv1 ≥ Kv. Luego, se volverá a dimensionar la estrategia PD con β = β2 y, posteriormente, la estrategia PI también con β = β2.

Lo siguiente, sería determinar el valor de la ganancia K_R del controlador para que los polos dominantes en lazo cerrado estén lo más cerca posible de los polos obtenidos mediante la estrategia PD.

Por último, se ha de evaluar la respuesta del sistema ante una entrada escalón unitario y calcular el valor de la constante de error, verificando que se cumplan las especificaciones. En caso de incumplimiento, se ajustarán los parámetros del controlador.

3.3.4. Método de respuesta en frecuencia

También se ha implementado el método de respuesta en frecuencia o método de Bode. La técnica empleada para la creación de reguladores continuos utilizando el enfoque de la respuesta en frecuencia se realiza de manera indirecta. A diferencia del método del lugar de las raíces, en este caso no se establecen directamente los polos en bucle cerrado. A través de este método, las tres formas de hacer que un sistema cumpla los requisitos por medio de un regulador es mediante la compensación adelanto de fase, la compensación atraso de fase y la compensación adelanto-atraso.

1. Red adelanto:

Determinar el valor de la ganancia del sistema en bucle abierto de manera que se cumplan las especificaciones del régimen permanente (errores constantes).

Construir el diagrama de Bode con la ganancia inicial del sistema sin compensar y calcular el margen de fase γ0 correspondiente.

Calcular el ángulo de avance de fase ϕc = ϕm necesario para complementar la curva de ángulo de fase del sistema sin compensar y cumplir con las especificaciones del margen de fase deseado, considerando un margen de seguridad adecuado (10 %).

Establecer el valor del parámetro α mediante la ecuación.

Calcular la frecuencia en la cual la curva de amplitud del sistema sin compensar es igual a $-20 \log \sqrt{1/\alpha}$, y considerar esta frecuencia como la nueva frecuencia de cruce de ganancia wgk, equivalente a la wm de la red de avance de fase donde ocurre el desfase máximo ϕm.

Calcular las frecuencias de corte $w1 = 1/T$ y $w2 = 1/(\alpha T)$ de la red de avance de fase. Para esto, es necesario calcular previamente T utilizando la ecuación.

Representar la respuesta en frecuencia del sistema compensado y verificar el margen de fase resultante. Si no se alcanza el valor deseado, repetir el diseño utilizando los valores obtenidos, pero con pequeñas modificaciones.

Aumentar la ganancia A del amplificador en $1/\alpha$ para tener en cuenta la atenuación de la red en bajas frecuencias. [4]

2. Red atraso:

El proceso a seguir para calcular las dimensiones de un controlador proporcional-integral (PI) o una red de retraso de fase es el siguiente:

Determinar el valor de la ganancia en lazo abierto que cumpla con las especificaciones de régimen estacionario (constantes de error).

Graficar el diagrama de Bode del sistema con la nueva ganancia calculada en el paso 1 y determinar el ángulo de fase de margen γ0 correspondiente.

Si las especificaciones no se cumplen, calcular la frecuencia wgk para la cual el ángulo de fase del sistema no compensado sea: $\gamma = -180^\circ + \gamma_{deseado}$, añadiendo 10° como margen de seguridad.

Establecer la frecuencia de corte $w2 = 1/T$ una década por debajo de la frecuencia wgk.

Calcular la atenuación necesaria a la frecuencia wgk para que la curva de ganancia |G(jw)| se intersecte con la línea de 0 dB en dicha frecuencia.

Calcular el valor del parámetro β considerando que la atenuación obtenida en el paso anterior debe ser $-20 \log \beta$.

Calcular la segunda frecuencia de corte, w1 = 1/(βT). [4]

3. Red adelanto-atraso:

La red de compensación de fase mediante retardo y avance aprovecha los beneficios de ambos tipos de redes, eliminando algunas de sus limitaciones. Su implementación es justificada en las siguientes situaciones: cuando las prestaciones deseadas no se pueden lograr utilizando solo una red, o cuando las especificaciones son demasiado exigentes y requieren una compensación mediante redes más complejas.

No existe un enfoque preciso y riguroso para dimensionar este tipo de reguladores, pero se pueden obtener resultados satisfactorios mediante un método de prueba y ajuste. Se combinarán las técnicas de diseño mencionadas anteriormente para cada una de las redes, asegurando que, durante la etapa de dimensionamiento, el parámetro α de la red de avance de fase sea igual al inverso del valor del parámetro β de la red de retardo de fase (α = 1/β). Siguiendo estos principios, la parte de retardo de fase de la red de compensación de fase proporcionará suficiente atenuación en las frecuencias cercanas a la frecuencia de cruce de ganancia (wgk). Esto permitirá aumentar la ganancia en la región de bajas frecuencias y, por lo tanto, mejorar el comportamiento en estado estacionario. La parte de avance de fase de la red de compensación de fase suministrará un ángulo de fase adicional en la zona de la frecuencia de cruce de ganancia (wgk), con el objetivo de mejorar el margen de fase. [4]

3.4. Otros cuadernos de análisis de sistemas

Como complemento a los métodos, se han diseñado otros cuadernos que sirven para el análisis y estudio de sistemas continuos. Destacamos el cuaderno LDR_Manual. Como se aprecia en la ilustración 27, nos permite visualizar la evolución del lugar de las raíces en función de los valores de k.

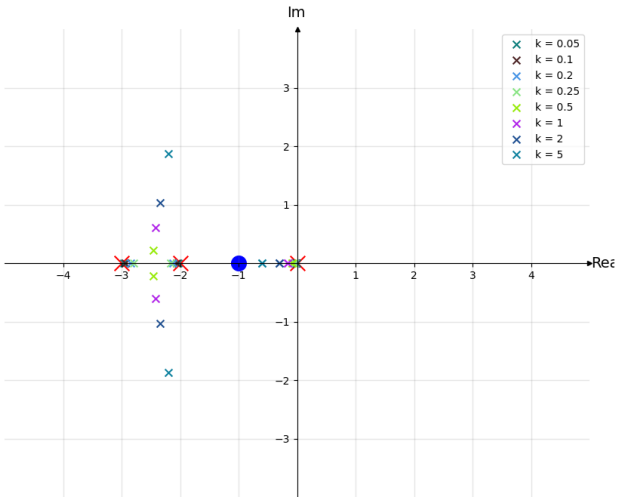

Ilustración 27. LDR Manual en función de k

En la ilustración 28 se muestran las regiones válidas del sistema.

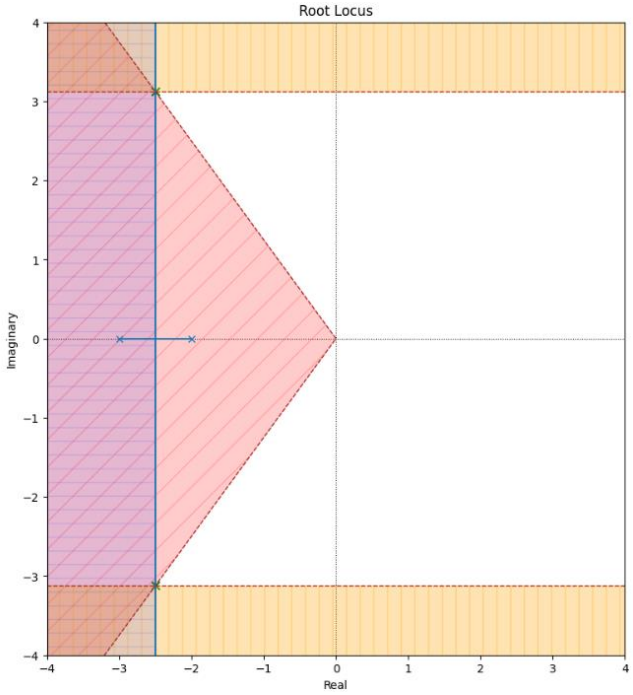

Ilustración 28. Regiones

4. Resultados

En este capítulo se mostrarán los resultados de los métodos y herramientas implementadas.

4.1. Método Manual

Como se ha expuesto en los anteriores capítulos el método manual, es un método introductorio que puede servir para diseñar el controlador a ojo viendo los cambios de la respuesta en función de los parámetros del controlador o como herramienta para comprobar los resultados obtenidos en cualquiera de los otros métodos.

En las ilustraciones 29 y 30, se muestran las salidas que nos da este cuaderno para el sistema con H=1 y $G_p = \frac{1}{s*(s+1)}$ y las condiciones iniciales siguientes: $e_p \leq 10\%, t_r \leq 0.5s, t_s \leq 1.5s, M_p \leq 15\%$ y $t_p \leq 3s$

```
ep_cond=10
tr_cond=0.5
ts_cond=1.5
Mp_cond=15
tp_cond=3
metodo_manual_param(ep_cond,tr_cond,ts_cond,Mp_cond,tp_cond,planta,sensor)
```

kp 12.26

ki 3.29

kd 5.23

```
 Parametros del sistema controlado
ep=7.541478129713424
tr=0.4242424242424242
ts=1.8383838383838382
Mp=10.650624576683219
tp=0.7070707070707071
M=
   5.23 s^2 + 12.26 s + 3.29
------------------------------
s^3 + 6.23 s^2 + 12.26 s + 3.29

El sistema no cumple la condición del tiempo de establecimiento menor o igual que 1.5
```

Ilustración 29. Ejemplo datos salida método manual

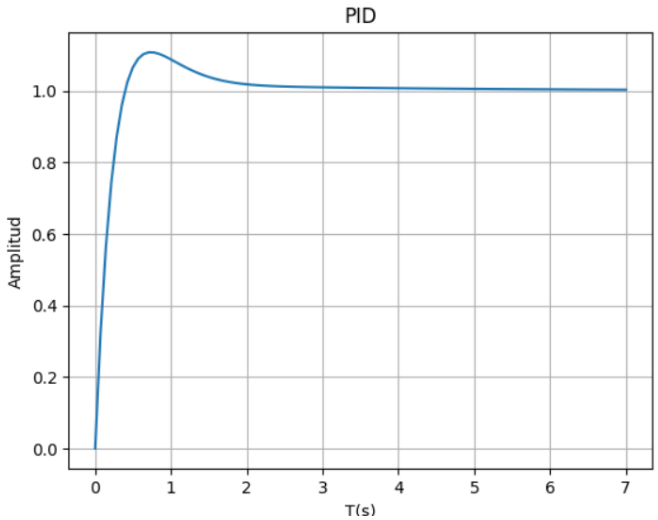

Ilustración 30. Gráfica de salida del sistema compensado con método manual

4.2. Métodos de diseño

4.2.1. Método fuerza bruta

El método de fuerza bruta, da como resultado una tabla, con los parámetros del sistema en función de cada combinación de valores de los parámetros del regulador.

Se muestran la salida que nos da este cuaderno (Tabla 5) con sólo los valores válidos para el sistema con H=1 y $G_p = \frac{1}{(s^3+3s^2+2)}$ y las condiciones iniciales siguientes: $e_p \leq 10\%, t_r \leq 0.5s, t_s \leq 1.5s, M_p \leq 15\%$ y $t_p \leq 1s$.

```
ep_cond=10
tr_cond=0.5
ts_cond=1.5
Mp_cond=15
tp_cond=1
G=funciones.lti([1],[1,3,2])
H=funciones.lti([1],[1])
salto=1
[kd,kp,ki,tr,ts,Mp,tp,ep]=metodo_fuerza_bruta_v3(ep_cond,tr_cond,ts_cond,Mp
_cond,tp_cond,vector_kp,vector_kd,vector_ki,G,H)
headers=["kd", "kp", "ki", "tr", "ts", "Mp", "tp", "ep"]
x=[]
for i in np.arange(0,len(kd),1):
  if ((tr[i]>tr_cond) or (ts[i]>ts_cond) or (Mp[i]>Mp_cond) or (tp[i]>tp_cond) or
(ep[i]>ep_cond)):
    continue
  else:
    x.append([kd[i],kp[i],ki[i],tr[i],ts[i],Mp[i],tp[i],ep[i]])
print(tabulate(x,headers))
```

Tabla 5. Ejemplo tabla de resultados método fuerza bruta

kd	kp	ki	tr	ts	Mp	tp	ep
3.1	18.1	15.1	0.490438	1.40125	10.9085	0.770689	5.24
3.1	18.1	17.1	0.469938	1.40981	12.508	0.763649	5.24
3.1	18.1	18.1	0.486209	1.45863	13.2802	0.756326	5.24
3.1	18.1	19.1	0.498146	1.44462	14.0288	0.797033	5.24
3.1	18.1	20.1	0.460538	1.47372	14.8492	0.782914	5.24
3.1	19.1	15.1	0.458226	1.29831	11.177	0.76371	4.98
3.1	19.1	16.1	0.490816	1.33221	11.9007	0.771282	4.98
3.1	19.1	17.1	0.452132	1.3564	12.6649	0.710493	4.98
3.1	19.1	18.1	0.477412	1.37256	13.4314	0.716118	4.98
3.1	19.1	19.1	0.442282	1.38213	14.1745	0.718708	4.98
3.1	19.1	20.1	0.462114	1.38634	14.899	0.718843	4.98
3.1	20.1	13.1	0.492078	1.18099	10.1603	0.68891	4.74
3.1	20.1	14.1	0.449298	1.25804	10.8934	0.718877	4.74
3.1	20.1	15.1	0.494573	1.23643	11.5153	0.741859	4.74
3.1	20.1	16.1	0.455494	1.29057	12.2525	0.683242	4.74
3.1	20.1	17.1	0.491135	1.26292	13.0316	0.701622	4.74
3.1	20.1	18.1	0.455318	1.30091	13.7446	0.7155	4.74
3.1	20.1	19.1	0.483755	1.33033	14.4268	0.725633	4.74
4.1	19.1	15.1	0.489979	1.32994	6.64084	0.769968	4.98
4.1	19.1	18.1	0.466944	1.45272	8.6149	0.778241	4.98
4.1	20.1	16.1	0.490434	1.33118	7.54213	0.700621	4.74
4.1	20.1	18.1	0.46821	1.40463	8.79053	0.760841	4.74
4.1	20.1	19.1	0.482133	1.39283	9.42909	0.749985	4.74
4.1	20.1	20.1	0.441429	1.42238	10.0445	0.735715	4.74
5.1	20.1	16.1	0.499014	1.30991	4.23702	0.748521	4.74
5.1	20.1	17.1	0.498032	1.38342	4.81754	0.774717	4.74
5.1	20.1	18.1	0.485129	1.45539	5.36836	0.776206	4.74
5.1	20.1	19.1	0.496739	1.49022	5.91692	0.786503	4.74

4.2.2. Método del lugar de las raíces

Como se ha indicado en el capítulo 3, el método del lugar de las raíces puede requerir más o menos pasos en función de las especiaciones del sistema. Para ello se ha creado una función denominada valorar_caso_LDR que indique en que caso estamos:

```
def valorar_caso_LDR(planta,ceros_planta,polos_planta):
 w=10000
 q=0
 for q in range(len(polos_planta)):
  if polos_planta[q]<w:
   w=polos_planta[q]
 dis_w=abs(w.real)
 #Si la distancia de un polo y un cero es < dis_w/6 --> Se pueden anular
 for e in range(len(ceros_planta)):
  for z in range(len(polos_planta)):
   if (abs(ceros_planta[e].real-polos_planta[z].real)) < dis_w/6:
    print('El cero:',ceros_planta[e],'y el polo:',polos_planta[z],'se pueden
anular')
   if (polos_planta[z].real) > dis_w*10:
    print('El polo:',polos_planta[z],'se puede anular')
 #Si la distancia de un polo es > dis_w*10 --> Se pueden anular
 fig,ax = funciones.plt.subplots(figsize=(10, 10))
 [rlist,klist]=LDR_Automático(ax,ceros_planta,polos_planta,5,5)
 i=0
 j=0
 o=[]
 #h=np.empty(shape=len(rlist)*2)
 h=np.ndarray(len(rlist)*2, dtype=np.complex128)
 r=0
 for i in range(len(rlist)):
  #u=list()
  u=rlist[i]
  o=np.round(u,2)
  h[j]=o[0]
  h[j+1]=o[1]
  #o[j+1]=list(round(u[i+1],2))
  #o[j]=(round(u[i].real,2),round(u[i].imag,2))
  #o[j+1]=complex(round(u[i+1].real,2),round(u[i+1].imag,2))
  j=j+2
 for r in range(len(h)):
  s=h[0]
  if(h[r].real>s):
   s=h[r]
 return w,dis_w
```

En la ilustración 31, se puede ver como la función indica que se pueden anular un par polo-cero. Por lo que se puede reducir el sistema.

```
ceros_planta=[-2]
polos_planta=[-0.43,-1.976,-3]
Ganancia_planta=0.05
Gp=lti(ceros_planta,polos_planta,Ganancia_planta)
#h,x=metodo_basico_basado_LDR(Gp,[15,1.5,10])
w,dis_w=valorar_caso_LDR(Gp,ceros_planta,polos_planta)
```

```
El cero: -2 y el polo: -1.976 se pueden anular
```

Ilustración 29. Ejemplo valorar_caso_LDR

Para un sistema $G_p = \frac{1}{(s+1)(s+2)}$, $M_p \leq 15\%,\ ts \cong 1.5\ y\ e_p \leq 10\%$

Se obtiene Gr como el que muestra la ilustración 32 y su respuesta (Ilustración 33).

```
Gr
```

$$\frac{13.46\,(s+0.35)\,(s+2)}{(s+0.16)\,(s+3.2)}$$

Ilustración 30. Controlador con LDR

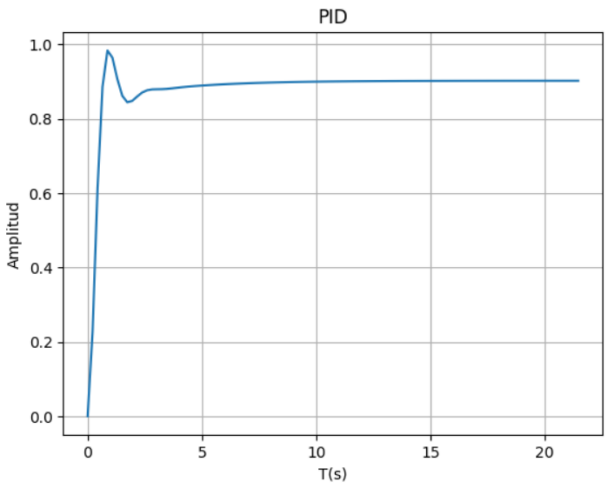

Ilustración 31. Respuesta del sistema controlado

4.2.3. Método de respuesta en frecuencia

Para un sistema $G_p = \frac{s+2}{(s+0{,}1)(s^2+10s+29)}$, $e_{rp} \leq 1\%$, $y\ \gamma_0 = 60$, se muestran los diagramas de bode en la ilustración 34.

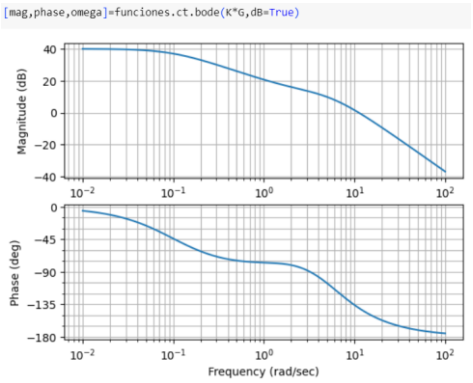

Ilustración 32. Diagramas de bode

4.3. Método de Ziegler-Nichols

4.3.1. Caso 1

Este caso es aplicable para sistemas que presenten la estructura:

$$G_P(s) = \frac{Ke^{-Ls}}{\tau s + 1}$$

Como se figura en la ilustración 35, este sería el resultado del controlador para una planta del tipo:

$$G_P(s) = \frac{2e^{-2s}}{9s + 1}$$

```
def diseño_ziegler_nichols_I(K,T,L):
  s = ct.tf('s')
  Kp=1.2*T/(K*L)
  Ti=2*L
  Td=0.5*L
  Gr=0.6*T*(s+1/L)*(s+1/L)/s
  return Gr,Kp,Ti,Td
```

```
Gr,Kp,Ti,Td=diseño_ziegler_nichols_I(2,9,2)
```

```
Gr
```

$$\frac{5.4s^2+5.4s+1.35}{s}$$

Ilustración 33. Ejemplo aplicación Ziegler-Nichols I

4.3.2. Caso 2

En la Ilustración 36 se muestran los resultados para un sistema con una planta $G_p = \frac{1}{s(s+1)(s+3)}$ y H=1

```
def diseño_ziegler_nichols_II(Kc,Pc):
  s = ct.tf('s')
  Kp=0.6*Kc
  Ti=0.5*Pc
  Td=0.125*Pc
  Gr=Kp*(1+1/(Ti*(s))+Td*(s))
  #Gr=0.6*Kp*(1+1/(*s)+Td*s)
  return Gr,Kp,Ti,Td
```

```
[Gr,Kp,Ti,Td]=diseño_ziegler_nichols_II(Kc,Pc)
```

```
Gr
```

$$\frac{793.8s^2+151.2s+7.2}{21s}$$

Ilustración 34. Ziegler Nichols II

5. Conclusiones

5.1. Conclusiones

En este trabajo de investigación, se ha desarrollado una aplicación interactiva en Jupyter Notebook utilizando el lenguaje de programación Python, con el objetivo de analizar sistemas continuos y diseñar controladores PID según las condiciones requeridas. Mediante el uso de herramientas interactivas, se ha logrado visualizar la información del sistema y evaluar el comportamiento del controlador diseñado, permitiendo realizar ajustes y verificar que se cumplen los requisitos establecidos.

El enfoque principal de este trabajo fue ir más allá de un diseño simple y básico de un controlador PID, buscando generar innovación y aportar un valor adicional al campo de los sistemas de control. La aplicación desarrollada permite al usuario explorar y comparar diferentes métodos de diseño, analizar el comportamiento del sistema en tiempo real y realizar ajustes de forma interactiva. Esto brinda una experiencia de aprendizaje, experimentación y mejora de los sistemas de control más enriquecedora y completa.

Durante el desarrollo de este proyecto, se ha realizado un análisis exhaustivo de investigación sobre los diversos métodos de diseño de controladores PID, haciendo especial hincapié en aquellos que, por su naturaleza, se han implementado en programas Python dentro de la aplicación interactiva. Estos programas, que bajo las condiciones dadas de las funciones de transferencia G(s) y H(s) del sistema, así como de los parámetros de diseño proporcionados, guían al usuario de forma interactiva y visual a través de los pasos necesarios para implementar el regulador. Además, aseguran que cumplen los requisitos tanto en el régimen transitorio como en el régimen permanente.

Concretamente se han implementado cinco cuadernos que corresponden cada uno a un método concreto: el método manual, el método de fuerza bruta, el método básico basado en el LDR con sus diferentes casos, los métodos de Ziegler-Nichols y el método de respuesta en frecuencia. Por último, se ha desarrollado un cuaderno final que evalúa todas las posibilidades para buscar el controlador más óptimo.

Si bien el desarrollo de este proyecto ha sido una experiencia entusiasta, no ha estado exenta de dificultades que han surgido en el transcurso del mismo. En primer lugar, el diseño de controladores PID en sistemas continuos es un tema complejo que involucra conceptos teóricos avanzados de teoría de control. Aplicar todos estos conceptos a la programación para que el análisis de sistemas y el diseño de controladores PID sea automático y directo, requiere un dominio amplio en la materia para poder implementar las herramientas y evaluar los resultados obtenidos. También, como se ha expuesto en el capítulo 3.1, implantar con la metodología de Jupyter Notebook en el entorno de Google Colab, presenta sus limitaciones que se han debido

considerar exhustivamente para depurar errores y optimizar el rendimiento de los programas.

En conclusión, en este trabajo se ha logrado desarrollar una aplicación interactiva en Jupyter Notebook basada en Python, que proporciona a los usuarios herramientas eficaces para el análisis y diseño de controladores PID en sistemas continuos. La combinación de la teoría explicada en el estado del arte con la implementación práctica en la aplicación interactiva ha permitido una comprensión más profunda de los conceptos y ha facilitado la experimentación y mejora de los sistemas de control.

A través de estas herramientas, se ha logrado realizar un estudio exhaustivo de los métodos de diseño de controladores PID, permitiendo al usuario explorar y comparar diferentes enfoques para lograr los objetivos deseados. La visualización de la información del sistema, junto con el controlador diseñado, ha proporcionado una retroalimentación en tiempo real que ha permitido verificar el cumplimiento de los requisitos establecidos y realizar ajustes precisos.

Además, se ha destacado la relevancia de utilizar Jupyter Notebook y Python como herramientas para el desarrollo de aplicaciones interactivas en el campo de los sistemas de control. Estas tecnologías ofrecen ventajas significativas, como la accesibilidad en la nube, la escalabilidad de los recursos computacionales, la integración con otras plataformas y la facilidad de colaboración, que han enriquecido la experiencia del usuario y han facilitado el aprendizaje y la experimentación en el ámbito de los sistemas de control.

El presente trabajo no solo ha cumplido con los objetivos planteados, sino que también ha sentado las bases para futuras investigaciones y desarrollos en el campo de los controladores PID. Se ha demostrado el potencial de las herramientas interactivas y la integración de la teoría y la práctica para lograr resultados más sólidos y efectivos en el diseño de sistemas de control.

5.2. Trabajos futuros

Como trabajos futuros y líneas de investigación que podrían abordarse para continuar avanzando en el desarrollo y mejora de la aplicación interactiva para el diseño de controladores PID, se propone lo siguiente:

- Ampliación de herramientas y métodos: Se puede considerar la expansión de los programas actuales para incluir más herramientas y métodos de diseño de controladores. Esto podría implicar la implementación de algoritmos más avanzados, como controladores PID fraccionales, controladores basados en técnicas de optimización, o controladores adaptativos, entre otros. El enfoque debe ser proporcionar a los usuarios una mayor variedad de opciones y técnicas para el diseño de controladores en sistemas continuos.

- Diseño de controladores en tiempo discreto: Una línea de investigación interesante sería el desarrollo de herramientas y métodos para el diseño de

controladores PID en sistemas de tiempo discreto. Esto permitiría abordar sistemas en los que el muestreo ocurre en intervalos discretos de tiempo, lo cual es común en aplicaciones digitales y de control en tiempo real. Sería necesario investigar y adaptar los métodos existentes al contexto de tiempo discreto, así como desarrollar nuevas técnicas específicas para este tipo de sistemas.

- Validación experimental: Una dirección prometedora sería la validación experimental de los controladores diseñados utilizando la aplicación interactiva. Esto implicaría implementar y probar los controladores PID en sistemas reales para evaluar su desempeño y compararlo con los resultados teóricos y de simulación. La experimentación práctica permitiría obtener retroalimentación sobre la eficacia de los controladores e identificar posibles mejoras o ajustes.

- Incorporación de técnicas de aprendizaje automático: Con el auge de la inteligencia artificial y el aprendizaje automático, sería interesante explorar la integración de estas técnicas en el diseño de controladores PID. Se podría investigar cómo utilizar algoritmos de aprendizaje automático para ajustar automáticamente los parámetros del controlador, adaptándolo a las características específicas del sistema y mejorando su desempeño.

- Optimización de controladores: Otra línea de investigación podría ser la optimización de los controladores PID para lograr un rendimiento óptimo en términos de criterios específicos, como el tiempo de respuesta, la estabilidad o el consumo de energía. Esto implicaría aplicar técnicas de optimización matemática y algoritmos evolutivos para encontrar los mejores valores de los parámetros del controlador y mejorar su eficiencia.

6. Bibliografía

[1] Ogata, K. (3ª Ed.). (1998). *Ingeniería de control moderna.* Prentice Hall.

[2] Kuo, B. C. (7ª Ed.). (1996). *Sistemas de control automático.* Prentice Hall.

[3] Castaño Giraldo, S.A. (23 de julio 2019). *Todo sobre Ziegler Nichols - Sintonía de Control PID*. Control Automático Educación. https://controlautomaticoeducacion.com/control-realimentado/ziegler-nichols-sintonia-de-control-pid/

[4] Ñeco, R. P., Reinoso, O., García, N. y Aracil, R. (2003). *Apuntes de Sistemas de Control.* Editorial Club Universitario.

[5] Alfaro Ruiz, V. M. (2003). *Métodos de sintonización de controladores PID que operan como servomecanismos*. Revista Ingeniería de la Universidad de Costa Rica, 13(1,2), 13-29.

[6] *Diseño de sistemas de control. Sintonía de PIDs*. OpenCourseWare. https://ocw.ehu.eus/file.php/83/capitulo10_html/capitulo10.html#s10322

[7] Ruge, I. A., Alvis, M. A., *Aplicación de los algoritmos genéticos para el diseño de un controlador PID adaptativo.* Tecnura, vol. 13, núm. 25, 2009, pp. 81-87

[8] *SciPy User Guide*. SciPy documentation. https://docs.scipy.org/doc/scipy/tutorial/index.html#user-guide

[9] Villanueva Garza, C. A. (2004), *Control de posición para un manipulador robótico utilizando retroalimentación difusa visual basada en imagen con cámara fija.* TESIS Maestría en Ciencias en Sistemas Inteligentes. Instituto Tecnológico y de Estudios Superiores de Monterrey.

[10] *Python Control Systems Library 0.9.4*. https://python-control.readthedocs.io/en/0.9.4/

[11] García Martínez, Esperanza (2016). *Desarrollo de un controlador PID industrial de bajo coste mediante Raspberry PI para control de temperatura*. Trabajo de Fin de Grado, Universidad Politécnica de Valencia, España.

[12] Lara Viera, D. A. (2019). *Diseño de Aplicación para Formación en Controladores PID Integrados en PLC*. Trabajo de Fin de Máster, Universidad de Sevilla, España.

[13] Cano, J. L. (14 de octubre de 2019). *Teoría de control en Python con SciPy (II): Control PID*. Pybonacci. https://pybonacci.org/2013/11/06/teoria-de-control-en-python-con-scipy-ii-control-pid/

[14] *Diseño de sistemas de control. Sintonía de PIDs*. OpenCourseWare. https://ocw.ehu.eus/file.php/83/capitulo10_html/capitulo10.html#s10322