



HERRAMIENTAS PARA EL ANÁLISIS Y DISEÑO DE CONTROLADORES EN GEOGEBRA

Autor: Cristofer Isaza Arenas

Director:
JOSÉ TOMAS BUITRAGO MOLINA, M. Sc.

FACULTAD DE INGENIERÍA ESCUELA DE INGENIERÍA MECÁNICA
SANTIAGO DE CALI, JUNIO DE 2025

Firma de Avalado por el director del proyecto:



A handwritten signature in black ink is written over a horizontal line. The signature is stylized, featuring a large loop at the top and a vertical stroke that descends below the line.

ÍNDICE DE CONTENIDO

RESUMEN	1
1. INTRODUCCIÓN	2
2. OBJETIVOS	3
2.1. Objetivo general	3
2.2. Objetivos específicos	3
3. PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN	4
4. ESTADO DEL ARTE	7
5. MARCO TEÓRICO	9
5.1. Dinámica de los sistemas de segundo orden	9
5.2. Variación del coeficiente de amortiguamiento	9
5.3. Variación de la frecuencia natural no amortiguada	10
5.4. Respuesta de un sistema de segundo orden subamortiguado ante una entrada escalón	10
5.4.1. Función de transferencia de segundo orden:	10
5.4.2. Tiempo pico	13
5.4.3. Tiempo de establecimiento:	14
5.4.4. Máximo sobreimpulso	14
5.4.5. Método de fracciones parciales para obtener la respuesta temporal de un sistema	14
1. Polos reales simples	14
2. Polos reales múltiples	14
3. Polos complejos conjugados	15
5.5. Polos de los sistemas de segundo orden	15
5.6. Acción de control PID	16
5.6.1. Acción proporcional (P)	17
5.6.2. Acción integral (I)	19
5.6.3. Acción derivativa (D)	19
5.6.4. Combinación de las acciones del control PID	21
5.7. Diseño por asignación de polos	21
5.8. Lugar de raíces del controlador PID	24
5.8.1. ¿Qué es el lugar geométrico de las raíces?	24
5.8.2. Método del lugar geométrico de las raíces	25
5.8.3. Reglas de construcción del lugar geométrico de las raíces	27
Puntos especiales del LGR:	27
Ramas del LGR:	27
Simetría con el eje real del LGR en control:	28
Comportamiento asintótico del LGR:	29
Cruce en el eje real de las asíntotas:	29
Lugar de las raíces en el eje real:	30
Polos adyacentes en el eje real:	31
Ángulo de partida en raíces complejas	31
Cruce con el eje imaginario:	32

5.9. GeoGebra	32
5.9.1. Guiones GeoGebra Script.....	32
5.9.2. Lenguajes de guiones.....	33
5.9.3. Botones	34
5.9.4. Cajas de entrada.....	34
5.9.5. Cajas de entrada vinculadas.....	34
5.9.6. Cajas vinculadas en construcciones.....	34
5.9.7. JavaScript en GeoGebra	35
6. METODOLOGÍA.....	37
6.1. Método de asignación de polos:.....	37
6.1.1 Procesamiento del frontend	40
6.1.2 Procesamiento del backend	45
6.2. Método del lugar geométrico de las raíces:.....	50
6.2.1 Procesamiento del frontend	52
7. RESULTADOS OBTENIDOS.....	56
7.1. Resultados del método de asignación de polos	56
7.2. Resultados del método del lugar geométrico de las raíces:.....	59
8. DISCUSIÓN DE RESULTADOS.....	62
9. CONCLUSIONES.....	64
10. IMPACTO DEL PROYECTO	65
11. RESULTADOS DE APRENDIZAJE ALCANZADOS	65
12. SUGERENCIAS PARA TRABAJOS FUTUROS.....	66
13. REFERENCIAS	67

ÍNDICE DE FIGURAS

Figura 1: Árbol del problema, fuente propia	5
Figura 2: Árbol de objetivos, objetivos específicos y actividades para el cumplimiento de los objetivos.....	6
Figura 3: Variación del factor de amortiguamiento ζ , tomado de [20].....	9
Figura 4: Variación de la frecuencia natural no amortiguada, tomado de [20]	10
Figura 5: Diagrama de bloques de un sistema de segundo orden, tomado de [19].....	11
Figura 6: Gráfica temporal de la respuesta de un sistema de segundo orden subamortiguado, tomado de [20].....	13
Figura 7: Diagrama de polos y ceros de un sistema subamortiguado de segundo orden, tomado de [20]	16
Figura 8: Diagrama de bloques de lazo cerrado con acción de control $C(s)$, entrada $r(t)$, perturbaciones $p(t)$, planta $G(s)$ y salida $y(t)$, tomado de fuente propia	17
Figura 9: Diagrama de bloques de un sistema de lazo cerrado con su acción de control proporcional, acción integral y derivativa, tomado de [20]	17
Figura 10: Diagrama de bloques, acción proporcional K_p , tomado de [20].....	18
Figura 11: Diagrama de bloques de controlador con acción proporcional integral, tomado de [20].....	19
Figura 12: Diagrama de bloques controlador PID proporcional, integral, derivativo, tomado de [20]	20
Figura 13: Diagrama de bloques de un proceso con controlador PID convencional, tomado de [20]	21

Figura 14 : Diagrama de bloques de un proceso con controlador PID y filtro pasa bajos, tomado de [20]	22
Figura 15: Lugar geométrico de las raíces y movimiento de los polos para el control de la temperatura de un horno con un controlador PI, tomado de [20]	25
Figura 16: Ramas del LGR por donde se ubican los polos del sistema conforme se aumenta la ganancia proporcional, tomado de [20]	28
Figura 17: Simetría de los polos con respecto al eje real, tomado de [20]	28
Figura 18: Asíntotas de las 3 ramas del lugar geométrico de las raíces, tomado de [20]	30
Figura 19: Polos y ceros con componente sólo en el eje real, tomado de [20]	31
Figura 20: Interfaz de GeoGebra, pestañas para editar en GeoGebra Script y JavaScript, tomado de [22]	33
Figura 21: Ejemplo de la creación de cajas vinculadas en una construcción, tomado de [22]	35
Figura 22-1: Diagrama de la arquitectura del método propuesto.	38
Figura 22-2: Interfaz del frontend con la respuesta temporal y polos del sistema mostrados en la vista gráfica de GeoGebra de la función de transferencia $10/(s^2+2s+10)$	38
Figura 23: Diagrama de flujo para el caso de uso del método de asignación de polos.....	39
Figura 24: Código en Javascript del frontend para determinar el coeficiente de amortiguamiento, la frecuencia natural y la función de planta $G(s)$	41
Figura 25: a) Primera parte código en Javascript del cálculo de los parámetros k_c , t_i , t_d según el tipo de controlador.....	42
Figura 26: Segunda parte código en Javascript del cálculo de los parámetros k_c , t_i , t_d según el tipo de controlador.....	43
Figura 27: Código en Javascript para la visualización de la función de lazo cerrado y su normalización	44
Figura 28: Código en Javascript de la función de transferencia de lazo cerrado normalizada en GeoGebra, polos del sistema y llamado de función inversa_L	45
Figura 29: Resultados en consola del procesamiento de la función $H(s)$ desde el backend, descomposición de fracciones parciales, formación del sistema de ecuaciones igualando los coeficientes de la ecuación característica deseada, obtención de coeficientes de las fracciones parciales y respuesta temporal $H(t)$	46
Figura 30: Primera parte código en Python de la función inversa_L del backend que procesa la función de transferencia de lazo cerrado.	47
Figura 31: Segunda parte código en Python de la función inversa_L del backend para la igualación de coeficientes de las fracciones parciales y la ecuación característica deseada.....	48
Figura 32: Tercera parte código en Python de la función inversa_L de la solución al sistema de ecuaciones de los coeficientes de las fracciones parciales y respuesta temporal del sistema	49
Figura 33 : Diagrama de flujo de caso de uso para el método del lugar geométrico de las raíces	51
Figura 34: Función iniciarLGR en Javascript del frontend para el método LGR.....	52
Figura 35: Función actualizarC para formar la expresión del controlador en GeoGebra	53
Figura 36: a) Primera parte función esperarGrado que verifica la existencia del polinomio característico de lazo cerrado y calcula los polos del sistema.....	54
Figura 36: b) Segunda parte función esperarGrado que dibuja el lugar geométrico de las raíces y llama la función inversa_L	55
Figura 37: Caso a) Resultados del método propuesto de asignación de polos con su respuesta temporal, polos del sistema y condiciones de diseño en $T_{ss}=0,588$ seg $M_p=23\%$, Tolerancia 2% del mismo proceso aplicado en el artículo [23]	56
Figura 38: Caso 1) Resultados del método propuesto del LGR en la interfaz web, con un controlador PID robusto y parámetros $K_p=2,058$ $K_i=15,83$ $K_d=0,0669$, mismo sistema utilizado en la figura 20 del artículo [18].....	59

Figura 39: Caso 2) Resultados del método propuesto del LGR en la interfaz web con controlador PID (Z-N) y parámetros $K_p=4,44$ $K_i=30,4$ $K_d=0,063$, mismo sistema utilizado en la figura 20 del artículo [18].....	59
---	----

ÍNDICE DE TABLAS

Tabla 1: Comparación de respuestas temporales ante una entrada escalón para distintos tipos de controladores, tiempos de asentamiento y sobreimpulso entre método del artículo [23] y método propuesto de asignación de polos	57
Tabla 2: Resumen de parámetros entre el método del artículo [23] y el método propuesto de asignación de polos	58
Tabla 3: Comparación del lugar geométrico de las raíces y respuesta temporal del caso 1 para un controlador PID robusto entre el método del artículo [18] y el método propuesto del LGR	60
Tabla 4: Comparación del lugar geométrico de las raíces y respuesta temporal del caso 2 para el controlador PID Z-N entre el método del artículo [18] y el método propuesto del LGR	61

RESUMEN

En el presente trabajo se ha creado una herramienta interactiva en GeoGebra centrada en el diseño de controladores PID a través de la aplicación de dos métodos clásicos ampliamente utilizados en ingeniería de control: la asignación de polos y el lugar geométrico de las raíces (LGR), tomando como referencia funciones de transferencia extraídas de publicaciones científicas. Se ha utilizado el lenguaje JavaScript para desarrollar una interfaz gráfica y un backend en Python usando la biblioteca SymPy para llevar a cabo cálculos simbólicos debido a las restricciones en la comunicación con el entorno CAS de GeoGebra. La herramienta creada permite que el usuario introduzca condiciones de diseño como el tiempo de establecimiento y sobre impulso para ver en tiempo real la ubicación de los polos y la respuesta temporal del sistema usando el primer método. Con respecto al segundo método, el usuario puede ingresar los parámetros del controlador y ver en tiempo real cómo se mueven los polos a medida que se incrementa la ganancia del sistema y observar el LGR junto con su respuesta temporal. Los resultados fueron comparados con herramientas profesionales como MATLAB y SISOTOOL y mostraron una coincidencia significativa en las respuestas. Esta solución evidenció que GeoGebra puede ser utilizado de manera efectiva y versátil tanto para propósitos educativos como para las fases preliminares de diseño en sistemas de control aunque no fuera originalmente diseñado para aplicaciones en automatización.

Palabras clave—controladores PID, GeoGebra, asignación de polos, lugar geométrico de las raíces, sistemas de segundo orden, respuesta temporal, automatización, JavaScript, Python, frontend, backend, SymPy.

1. INTRODUCCIÓN

En la actualidad se prioriza la creación de entornos de aprendizaje dinámicos y accesibles en el ámbito académico mediante la integración de tecnologías digitales y herramientas matemáticas en el proceso educativo. En la última década ha aumentado el interés por parte de docentes de todos los niveles y comunidades de aprendizaje en todo el mundo hacia esta tendencia innovadora [1]. Dentro de este contexto, GeoGebra ha surgido como una de las plataformas más destacadas, ya que es un programa gratuito que funciona en diferentes sistemas operativos e integra el ámbito de la enseñanza de disciplinas STEM (Ciencia, Tecnología, Ingeniería y Matemáticas), atrayendo a una gran cantidad de usuarios alrededor del mundo. Se ha posicionado como uno de los principales proveedores de herramientas matemáticas dinámicas, por su capacidad para facilitar la visualización interactiva de conceptos, mejorar el rendimiento académico y fomentar un aprendizaje más significativo y motivador [2], tal como lo evidencia un estudio meta-analítico que encontró un impacto altamente positivo en las habilidades matemáticas de los estudiantes que utilizan Geogebra, con una magnitud del efecto de 1,31, y realza especialmente la comprensión conceptual de la materia, independientemente del nivel educativo [3].

El motor matemático de GeoGebra impulsa una gran cantidad de sitios educativos en todo el mundo que ofrecen demostraciones dinámicas e interactivas y sistemas avanzados de evaluación en línea. Un ejemplo de ello es la plataforma de recursos digitales del Gobierno de Canarias, que integra *applets* de GeoGebra en sus secciones de optimización, cálculo y probabilidad, permitiendo a los estudiantes interactuar con escenas dinámicas que ilustran conceptos como la derivada, la integral o la ley de los grandes números [4]. Desde que se incorporó al grupo BYJU en 2021, la plataforma sigue funcionando como una entidad independiente bajo la supervisión de sus fundadores originales y amplía su alcance para llegar a cientos de millones de estudiantes en plataformas educativas digitales [5]. Esta transformación coloca a GeoGebra no únicamente como una herramienta para la enseñanza, sino como un contexto de desarrollo versátil y efectivo que tiene la capacidad de ser utilizado también en campos más específicos como la ingeniería. Tal como lo hace el presente trabajo, el cual desarrolló una herramienta interactiva en GeoGebra centrada en el diseño de controladores PID a través de la aplicación de dos métodos clásicos del control y la intervención de lenguajes como Python y Javascript para automatizar el diseño y hacerlo dinámico al público.

Por otro lado, el área de automatización y control ha cobrado gran relevancia en los últimos años debido a su capacidad de mejorar la eficiencia, la precisión y la seguridad en procesos industriales, tecnológicos y domésticos. Esta disciplina se basa en el diseño de sistemas capaces de autorregularse mediante sensores y actuadores, aplicando acciones correctivas sobre variables fundamentales como temperatura, presión, velocidad o posición, con el objetivo de mantenerlas dentro de valores deseados [6]. Un ejemplo de su impacto es la implementación de bodegas totalmente automatizadas, como la desarrollada por Treasury Wine Estates en Barossa Valley (Australia), donde sistemas robotizados y software de control permiten aumentar la eficiencia de producción hasta en un 60 % y mejorar la trazabilidad y la seguridad en las operaciones [7]. En este contexto, los controladores PID (Proporcional, Integral y Derivativo) son ampliamente utilizados por su simplicidad, efectividad y adaptabilidad a una amplia gama de procesos. Un ejemplo de su aplicación es el control de temperatura en hornos industriales, donde los PID regulan de forma automática la potencia de calefacción para mantener variaciones de hasta $\pm 0,5$ °C, optimizando el consumo energético y asegurando la calidad del producto final [8].

Impulsado por estos avances recientes en tecnología, el objetivo principal de este proyecto es investigar las capacidades del entorno GeoGebra en el campo del control automático, específicamente en la aplicación de controladores PID en sistemas de segundo orden personalizados. Con este propósito, se ha desarrollado una herramienta interactiva programada en JavaScript que funciona en el entorno visual de GeoGebra y permite calcular automáticamente los parámetros del controlador utilizando métodos tradicionales como la asignación de polos y el lugar geométrico de las raíces (LGR). El núcleo del cálculo se basa en la ecuación de búsqueda derivada de los requisitos de diseño:

$$\zeta = \frac{\ln(M_p)}{\sqrt{\pi^2 + [\ln(M_p)]^2}}, \omega_n = \frac{4}{\zeta T_s}$$

donde M_p es el sobrepaso porcentual y T_s el tiempo de asentamiento especificado por el usuario. Estos valores permiten determinar los polos complejos deseados y, a partir de ellos, dimensionar el PID mediante las rutinas de análisis simbólico. La “base de datos” del sistema está conformada por las librerías simbólicas de GeoGebra y un backend desarrollado en Python con SymPy, que se activa automáticamente cuando el motor simbólico interno no es suficiente. El usuario simplemente ingresa los parámetros de diseño y la herramienta se encarga de mostrar gráficamente la respuesta del sistema, garantizando que se comporte como uno de segundo orden subamortiguado, lo que facilita la comprensión visual del impacto de los parámetros del controlador sobre la dinámica del sistema.

De este modo, el proyecto no solo demuestra la viabilidad de GeoGebra como una herramienta educativa en ingeniería, sino que también amplía sus posibilidades como plataforma de prototipado rápido para el diseño y validación de sistemas de control. La flexibilidad para integrarse con lenguajes externos como Python y su enfoque visual e interactivo convierten a GeoGebra en una opción valiosa para contextos académicos, de investigación y de bajo presupuesto, fomentando el aprendizaje práctico de conceptos avanzados en automatización.

3. PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

Las aplicaciones educativas son herramientas esenciales que ayudan a facilitar el aprendizaje de los conceptos enseñados en la universidad o en el aula escolar. Esto permite a los estudiantes que tienen acceso a internet aplicar de forma práctica lo que han aprendido en diversas áreas de estudio. Una de estas herramientas es GeoGebra; un software matemático dinámico que proporcionará entornos de geometría y álgebra para todos los niveles educativos en un solo paquete integrado que incluye hojas de cálculo y gráficos estadísticos junto a otras funciones esenciales de trigonometría, álgebra y estadística. Por esto, esta herramienta brinda una amplia gama de utilidades matemáticas que van más allá de ser simplemente una calculadora en línea; se convierte en una suite completa para el aprendizaje y la enseñanza matemática. Además de todo esto, GeoGebra proporciona una plataforma en línea que cuenta con más de 1 millón de recursos gratuitos disponibles para sus usuarios [9]. De este modo tanto los alumnos como los docentes pueden disfrutar de una innovadora aproximación a la enseñanza de las matemáticas sin necesidad de invertir dinero alguno y todo ello impulsado por este software gratuito disponible.

GeoGebra es una herramienta interactiva que se emplea para enriquecer la comprensión de las matemáticas en estudiantes de educación primaria y secundaria. Fue desarrollada por Markus Hohenwarter en 2002 y actualmente cuenta con un equipo de programadores dedicados a incorporar mejoras como una comunidad activa de colaboradores que comparten sus recursos y saberes a nivel global. Es importante mencionar que al ser una aplicación completamente gratuita se utiliza ampliamente en la comunidad estudiantil; no obstante tiene limitaciones cuando se necesita aplicar conceptos de matemáticas avanzadas en el campo de la automatización y el control. Dado que GeoGebra se enfoca en áreas como geometría, trigonometría y cálculo para desarrollar el pensamiento espacial y usar ambientes informáticos para funciones y conceptos básicos de educación financiera entre otros. No incluye herramientas de control como el cálculo de controladores PID. Estos son útiles para estabilizar el rendimiento de un sistema en tiempo real al reducir las oscilaciones y mejorar la precisión [10].

En los últimos tiempos la automatización ha cobrado gran relevancia en los procesos industriales al transformar la manera en que se desarrollan las actividades de producción y resulta crucial en la optimización de dichos procesos. Además de posibilitar una producción más exacta y disminuir los gastos de funcionamiento conllevando así a una mayor competitividad en el mercado. Sin embargo, a pesar de que el programa matemático GeoGebra es una herramienta de software de código abierto gratuito, ha sido intervenido muy poco en el ámbito de la automatización hasta ahora. Aunque tiene sus propios desarrolladores web, no se han producido muchos avances en el campo del cálculo de parámetros relacionados con esta área, como funciones de transferencia, controladores o herramientas similares. Esto puede deberse al desconocimiento del lenguaje en que opera el programa mismo. Debido al escaso interés de los programadores y propietarios en adentrarse en este ámbito tecnológico; no obstante, gracias a su interfaz y flexibilidad se pueden generar simulaciones interactivas que muestren de forma visual cómo los sistemas de control reaccionan ante diversas entradas.

Los gráficos y la animación son herramientas útiles tanto para estudiantes como para profesionales que desean comprender de manera más efectiva el comportamiento dinámico de sistemas controlados. Asimismo pueden ser empleados en la configuración de los controladores PID; por ejemplo permitiendo ajustar parámetros y observar en tiempo real cómo varía la respuesta del sistema. Además de todo esto facilitan la representación visual y manipulación de ecuaciones diferenciales y funciones de transferencia cuando se utilizan adecuadamente en cada entorno del programa ya sea algebraico o simbólico antes de pasar al aspecto gráfico correspondiente. Es por eso que destacar y mejorar este aspecto en un programa como GeoGebra puede tener muchos beneficios tanto para el ámbito educativo como para el empresarial. En el ámbito

educativo se pueden beneficiar los estudiantes al poder familiarizarse con las nuevas tecnologías utilizadas en la industria y así poder desarrollar propuestas innovadoras; mientras que para el sector empresarial esto supone una consecuencia positiva al acercarse a la automatización que resultará en una mejor gestión, dinamismo y avances en la industria, fomentando nuevas ideas y mejoras en sus procesos. Estas razones resumidas se enumeran en el árbol del problema que se muestra en la figura 1.

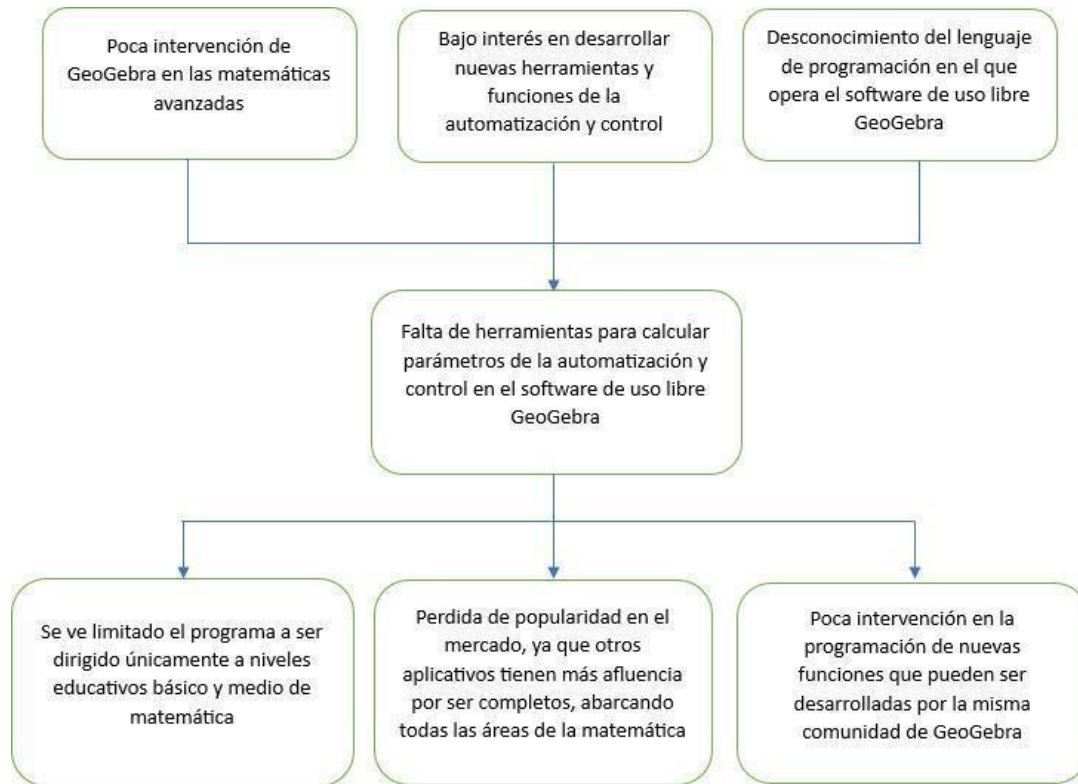


Figura 1: Árbol del problema, fuente propia

Una alternativa para esta problemática, se plantea en este proyecto, el cual pretende desarrollar una herramienta dentro de GeoGebra, utilizando su lenguaje interno (JavaScript) que incorpore funciones propias del análisis y diseño de sistemas de control, tales como: la visualización del lugar geométrico de las raíces, los polos del sistema, función de lazo cerrado, respuesta temporal, entre otros parámetros necesarios para el diseño de controladores PID. La propuesta busca aprovechar la precisión gráfica y el entorno interactivo del software para ofrecer al usuario una experiencia didáctica enriquecida, permitiéndole ingresar parámetros de diseño y observar, en tiempo real, cómo se comporta el sistema en función de los ajustes realizados.

Con ello, se espera no solo potenciar el uso de GeoGebra en contextos universitarios vinculados a la ingeniería de control, sino también promover un mayor acercamiento entre el ámbito académico y el industrial. La incorporación de estas herramientas facilita la comprensión de conceptos avanzados en automatización mediante una interfaz accesible, visual y adaptable, beneficiando tanto a estudiantes como a profesionales. En consecuencia, se plantea una solución que no solo atiende una limitación técnica, sino que también impulsa la innovación pedagógica y técnica en el área de control automático. Para lograr el correcto desarrollo del proyecto se desglosa cada objetivo, de manera que las actividades propuestas permitan el cumplimiento de cada uno de ellos, el desglose de cada objetivo se muestra en la figura 2.

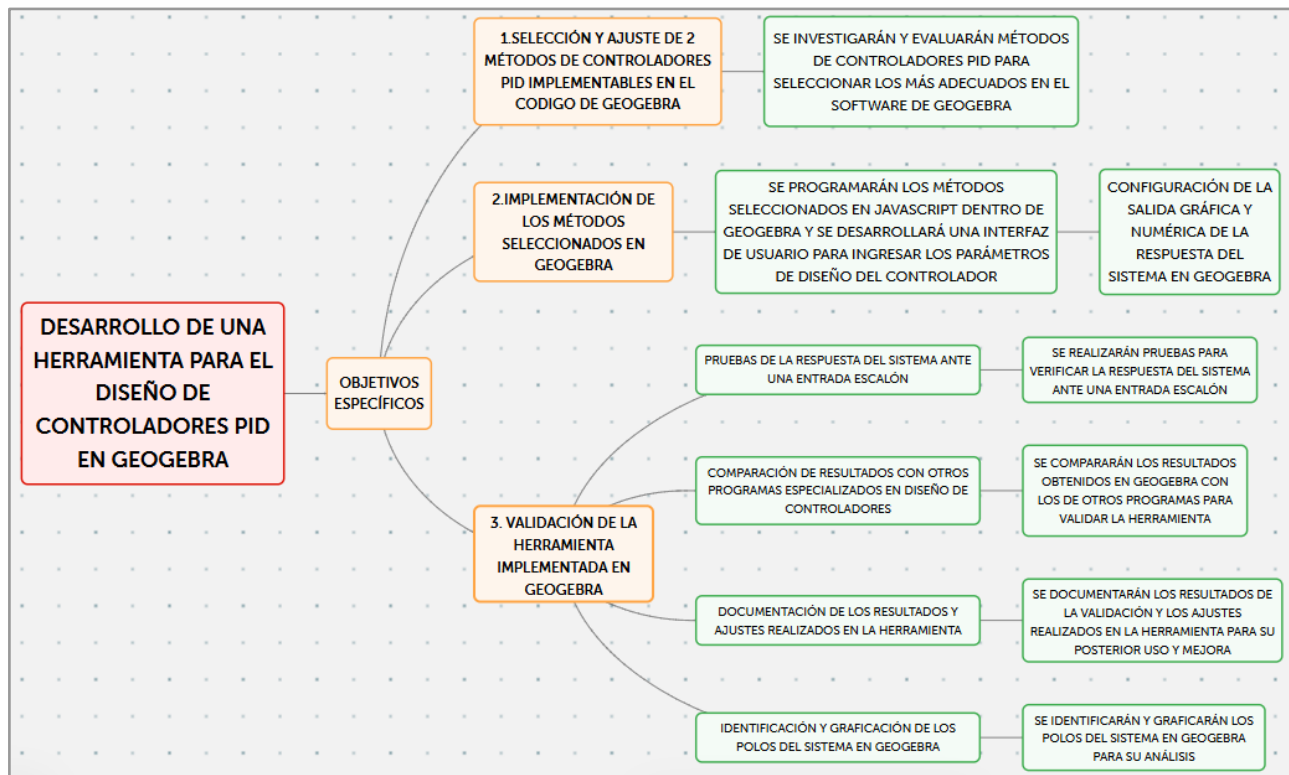


Figura 2: Árbol de objetivos, objetivos específicos y actividades para el cumplimiento de los objetivos.

2. OBJETIVOS

2.1. Objetivo general

Desarrollar una herramienta que encuentre los parámetros de un controlador PID ajustando el sistema a un sistema de segundo orden subamortiguado, utilizando el software libre de uso matemático conocido como GeoGebra y con una función de transferencia establecida

2.2. Objetivos específicos

- 2.2.1. Seleccionar, y ajustar 2 métodos de controladores PID, que sean implementables en el lenguaje de programación de GeoGebra, los cuales permiten tanto graficar cómo calcular parámetros importantes de los controladores.
- 2.2.2. Implementar en el software libre de uso matemático GeoGebra los 2 métodos de controladores PID seleccionados, ajustando la lógica de programación necesaria en el lenguaje Javascript de GeoGebra, para que el usuario sólo digite los parámetros iniciales de diseño del controlador y se obtenga la respuesta del sistema.
- 2.2.3. Validar la herramienta implementada en el software de uso matemático GeoGebra mostrando la respuesta gráfica del sistema ante una entrada escalón y sus polos correspondientes, comparando resultados con otros aplicativos especiales para diseño de controladores.

4. ESTADO DEL ARTE

GeoGebra ha alcanzado un gran nivel de popularidad en las aulas de matemática en gran parte del mundo. Uno de los factores clave para su permanencia en la escena educativa es que constantemente se está reinventando e incorporando nuevas posibilidades. En los últimos años, ha tenido un alto grado de desarrollo la vertiente tridimensional del programa, incorporando la realidad aumentada y la opción de exportar construcciones para imprimir en 3D. Esta vista se incorpora a partir de la versión 5 del programa. Permite la representación gráfica de distintos objetos en el espacio tridimensional: puntos, rectas, segmentos, polígonos, curvas, poliedros, superficies, gráficas de funciones de dos variables, etc. Asimismo permite la representación algebraica de muchos de estos objetos: curvas y superficies en forma paramétrica; esferas, planos y cuádricas en forma implícita; todo esto, dinámicamente conectado como ya es costumbre en GeoGebra. Esto posibilita el trabajo con múltiples representaciones en simultáneo [11].

Otro avance importante se publicó en marzo del 2023, en la cual se anuncia la integración del lenguaje de programación Python a la interfaz de GeoGebra para dinamizar y posibilitar la intervención directa en las funciones y procesos del aplicativo [12]. Esto es un gran avance ya que el lenguaje de programación Python según el índice TIOBE ocupa el primer lugar con un 16,2%, el cual es un indicador de la popularidad de los lenguajes de programación. El índice se actualiza una vez al mes y las calificaciones se basan en la cantidad de ingenieros capacitados en todo el mundo, cursos y proveedores externos. Sitios web populares como Google, Amazon, Wikipedia, Bing, entre otros, se utilizan para calcular las calificaciones [13].

Generalmente se evidencia los avances en el aplicativo GeoGebra en torno a la geometría, trigonometría, cálculo, el desarrollo del pensamiento espacial, ambientes informáticos como funciones, dominios de funciones vectoriales, conceptos básicos de educación financiera, simulaciones físicas, entre otras. Se realizó una búsqueda bibliográfica en las bases de datos Google Scholar con la ecuación GeoGebra + Automatización, sin embargo sólo se encontraron artículos como: Software GeoGebra en la mejora de capacidades resolutivas de problemas de figuras geométricas bidimensionales en universitarios, GeoGebra como herramienta didáctica para el fortalecimiento del aprendizaje de secciones cónicas en bachillerato, el uso del software GeoGebra en la resolución de sistemas de inecuaciones, aplicaciones de GeoGebra en la determinación de máximos y mínimos en línea, sólo el artículo más cercano a la automatización se trató de automatización del control del conocimiento matemático dentro de programas de matemática dinámica, la cual hace referencia a las formas de control de los conocimientos matemáticos sobre la base de los programas de matemáticas dinámicas y discuten la hipótesis, probada experimentalmente, sobre la ecuación estadística del tiempo necesario para la ejecución de tareas geométricas de construcción, de investigación y de determinación geométrica de puntos [14]. Cabe resaltar que es un aplicativo altamente gráfico y por ende plantea sus soluciones de esa manera. Un ejemplo de ello, se ve en la solución de problemas de programación lineal de dos variables, la cual puede llevarse a cabo a través del “método gráfico”, que implica la construcción de la “región factible (RF)” de donde se obtienen los valores de las variables que producen un valor óptimo (mínimo o máximo) de la función objetivo (FO).

Entre los recursos que exploran el uso de GeoGebra en combinación con controladores PID, se encuentra el control PID de un Péndulo Invertido. A través de GeoGebra, se pueden ajustar los parámetros del controlador PID y observar cómo estos afectan la estabilidad del sistema. Esta simulación es particularmente útil en entornos educativos para ilustrar cómo funciona un controlador PID y cómo se puede ajustar para lograr el control deseado del sistema [15]. Otro recurso en GeoGebra proporciona una

interfaz gráfica para simular un regulador PID. Los usuarios pueden modificar las constantes proporcionales, integrales y derivativas, y observar el impacto de estos cambios en un sistema simulado. Este tipo de herramientas es muy valioso para la educación en control automático, ya que permite a los estudiantes visualizar y comprender mejor los principios que subyacen en el control PID [16]

Otro avance, se encontró en la revista IEEE Xplore, la propuesta de un método de diseño gráfico interactivo, utilizando GeoGebra como herramienta para diseñar un controlador robusto. En la cual, GeoGebra permite ajustar el compensador, de modo que el gráfico de Nyquist en bucle abierto fuese tangencial al círculo de cierta sensibilidad característica en todas las frecuencias, mientras permanece fuera del círculo de sensibilidad dentro del ancho de banda designado. Con lo que se logró que la ganancia en el bucle cerrado se encontrara en un máximo de frecuencias, y se logró tanto límites de sensibilidad deseados como límites en la señal de entrada máxima de la planta [17].

Pocos son los avances encontrados de este aplicativo en el área de la automatización y control. Si bien el aplicativo tiene herramientas para visualizar gráficamente los resultados, como lo utiliza el trabajo de Yudha Firizky Dwitama, en la cual, se consideró que los estudiantes no entendían rápida y fácilmente el proceso de implementación de cálculo de controladores PID de los libros y la influencia de los parámetros K_p , K_i y K_d ; por lo que se creó una visualización en GeoGebra para mostrar los procesos que ocurren en el bloque de cálculo de error, la influencia de la ganancia propuesta en el bloque del controlador proporcional, la influencia de la ganancia en el bloque integral y la influencia de los cambios de error en el bloque del controlador derivativo[18]. Sin embargo, no existen funciones propias de GeoGebra dedicadas al área del control.

5. MARCO TEÓRICO

5.1. Dinámica de los sistemas de segundo orden

El comportamiento dinámico de un sistema de segundo orden puede ser descrito en términos de dos parámetros: ζ y ω_n .

Donde ω_n representa la frecuencia natural no amortiguada, es decir, la frecuencia a la que oscilaría el sistema si no tuviera amortiguamiento. Se mide en radianes por segundo (rad/s) y está directamente relacionada con la rapidez de la respuesta del sistema.

Donde ζ es el coeficiente de amortiguamiento y representa una medida adimensional del amortiguamiento relativo del sistema. Describe cómo se disipa la energía en el sistema oscilatorio. Dependiendo de su valor, el sistema puede presentar distintos comportamientos dinámicos[19]:

- $\zeta=0$ Sistema oscilatorio (oscila indefinidamente)
- $0<\zeta<1$ Sistema subamortiguado (oscilaciones que se atenúan con el tiempo)
- $\zeta=1$ Sistema críticamente amortiguado (sin oscilaciones y retorno más rápido al equilibrio)
- $\zeta>1$ Sistema sobreamortiguado (sin oscilaciones, pero más lento que el caso crítico)

5.2. Variación del coeficiente de amortiguamiento

La dinámica de sistemas de segundo orden ante la variación del coeficiente de amortiguamiento ζ se muestra en la figura 3 :

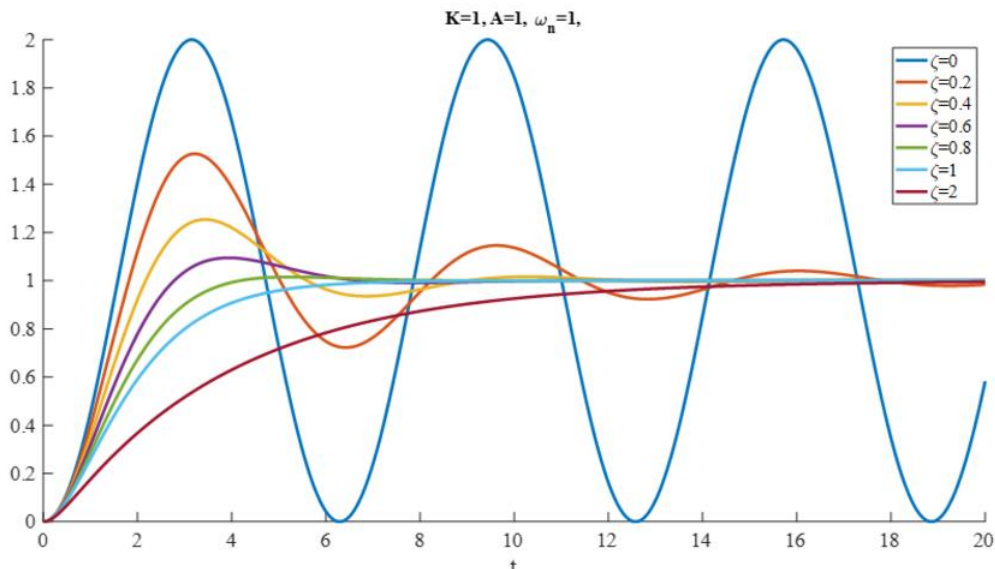


Figura 3: Variación del factor de amortiguamiento ζ , tomado de [20]

5.3. Variación de la frecuencia natural no amortiguada

El cambio de la frecuencia natural no amortiguada refleja un aumento o disminución en el periodo de oscilación de la respuesta temporal, cómo se observa en la figura 4.

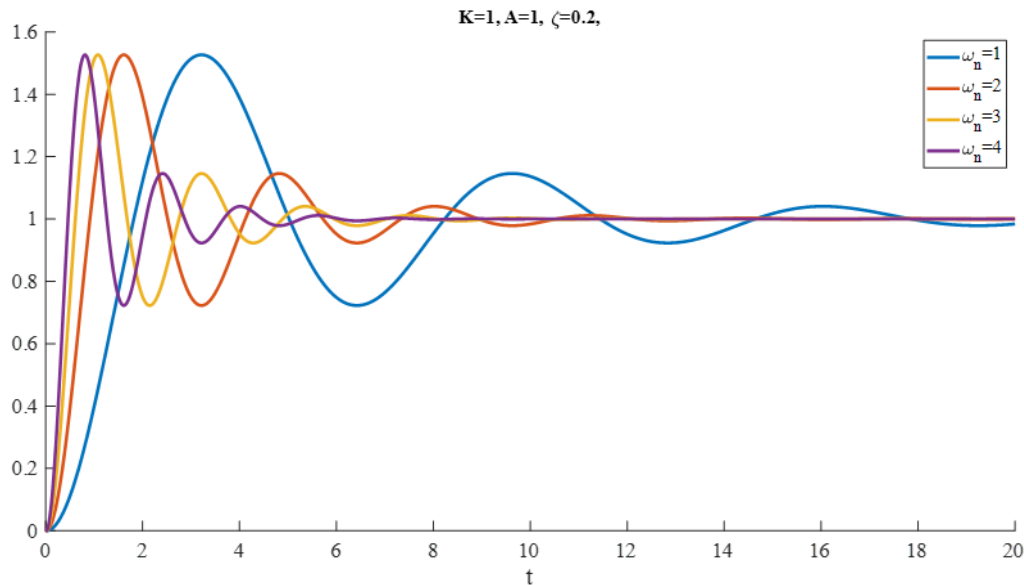


Figura 4: Variación de la frecuencia natural no amortiguada, tomado de [20]

5.4. Respuesta de un sistema de segundo orden subamortiguado ante una entrada escalón

5.4.1. Función de transferencia de segundo orden:

Un sistema de segundo orden tiene una función de transferencia de la forma:

$$\frac{Y(s)}{R(s)} = \frac{K \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (1)$$

Esta ecuación es llamada la forma estándar de la función de transferencia de un sistema de segundo orden, donde ζ está definido como la razón de amortiguación y es adimensional y ω_n representa la frecuencia natural del sistema. Su diagrama de bloques viene dado cómo el mostrado en la figura 5.

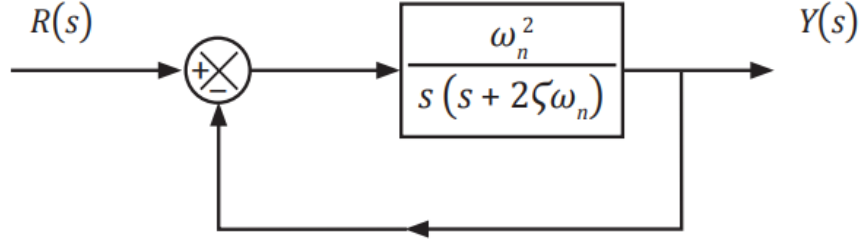


Figura 5: Diagrama de bloques de un sistema de segundo orden, tomado de [19]

La ecuación característica del modelo de segundo orden la representa el denominador de la ecuación (1) igualada a cero:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \quad (2)$$

La respuesta del sistema se obtiene reemplazando la entrada $R(s)$ por una entrada escalón de amplitud A :

$$Y(s) = \frac{AK \omega_n^2}{s \cdot (s^2 + 2\zeta\omega_n s + \omega_n^2)} \quad (3)$$

Si se considera los polos del sistema cómo polos complejos conjugados (cuando $0 < \zeta < 1$) cómo:

$$s_{1,2} = \alpha \pm j \beta \quad (4)$$

Al aplicar fracciones parciales a la ecuación 3 se obtiene:

$$Y(s) = \frac{D}{s} + B \left[\frac{(s-\alpha)}{(s-\alpha)^2 + \beta^2} \right] + \frac{B\alpha+C}{\beta} \left[\frac{\beta}{(s-\alpha)^2 + \beta^2} \right] \quad (5)$$

Teniendo en cuenta que $\alpha = -\zeta\omega_n$ y $\beta = \omega_d$ donde ζ es la constante de amortiguamiento, ω_n es la frecuencia natural y ω_d la frecuencia amortiguada del sistema. Entonces, la ecuación 5 toma la forma de:

$$Y(s) = \frac{D}{s} + B \left[\frac{(s+\zeta\omega_n)}{(s+\zeta\omega_n)^2 + \omega_d^2} \right] + \frac{-B\zeta\omega_n+C}{\omega_d} \left[\frac{\omega_d}{(s+\zeta\omega_n)^2 + \omega_d^2} \right] \quad (6)$$

Los coeficientes B , C y D se pueden hallar resolviendo la ecuación 6.

- **Coeficiente D:**

$$D = \lim_{s \rightarrow 0} \left[s \frac{AK\omega_n^2}{s \cdot (s^2 + 2\zeta\omega_n s + \omega_n^2)} \right] = AK \quad (7)$$

- **Coefficiente B:**

$$B = \frac{1}{\omega_d} \text{imag} \left[\lim_{s \rightarrow -\zeta\omega_n + j\omega_d} \left\{ [(s + \zeta\omega_n)^2 + \omega_d^2] \frac{AK\omega_n^2}{s \cdot (s^2 + 2\zeta\omega_n s + \omega_n^2)} \right\} \right]$$

$$B = \frac{1}{\omega_d} \text{imag} \left[\lim_{s \rightarrow -\zeta\omega_n + j\omega_d} \left\{ \frac{AK\omega_n^2}{s} \right\} \right]$$

$$B = \frac{1}{\omega_d} \text{imag} \left\{ \frac{AK\omega_n^2}{-\zeta\omega_n + j\omega_d} \right\}$$

Multiplicando por el conjugado:

$$B = \frac{1}{\omega_d} \text{imag} \left\{ \frac{AK\omega_n^2}{-\zeta\omega_n + j\omega_d} \cdot \frac{-\zeta\omega_n - j\omega_d}{-\zeta\omega_n - j\omega_d} \right\}$$

$$B = \frac{1}{\omega_d} \text{imag} \left\{ \frac{AK\zeta\omega_n^3 - jAK\omega_n^2\omega_d}{\zeta^2\omega_n^2 + \omega_d^2} \right\}$$

Recordando que $\omega_d = \sqrt{1 - \zeta^2}$

$$B = \frac{1}{\omega_d} \text{imag} \{ AK\zeta\omega_n - jAK\omega_d \}$$

$$B = \frac{1}{\omega_d} \{-AK\omega_d\} = -AK \quad (8)$$

- **Coefficiente $\frac{C - B\zeta\omega_n}{\omega_d}$:**

$$\frac{C - B\zeta\omega_n}{\omega_d} = \frac{1}{\omega_d} \text{Real} \{ AK\zeta\omega_n - jAK\omega_d \}$$

$$\frac{C - B\zeta\omega_n}{\omega_d} = \frac{1}{\omega_d} [-AK\zeta\omega_n] = \frac{-AK\zeta}{\sqrt{1 - \zeta^2}} \quad (9)$$

Reemplazando los coeficientes en la ecuación 6:

$$Y(s) = \frac{AK}{s} - AK \left[\frac{(s + \zeta\omega_n)}{(s + \zeta\omega_n)^2 + \omega_d^2} \right] - \frac{AK\zeta}{\sqrt{1 - \zeta^2}} \left[\frac{\omega_d}{(s + \zeta\omega_n)^2 + \omega_d^2} \right] \quad (10)$$

La salida en el dominio temporal, se obtiene con la transformada inversa de Laplace de $Y(s)$, el resultado es:

$$Y(t) = AK \cdot [1 - e^{-\zeta \omega_n t} (\frac{\zeta}{\sqrt{1-\zeta^2}} \text{sen}(\omega_d t) + \text{cos}(\omega_d t))] \quad (11)$$

Gráficamente la respuesta ante un escalón unitario de un sistema de segundo orden subamortiguado, cómo el presentado en la ecuación 4, se observa en la figura 6.

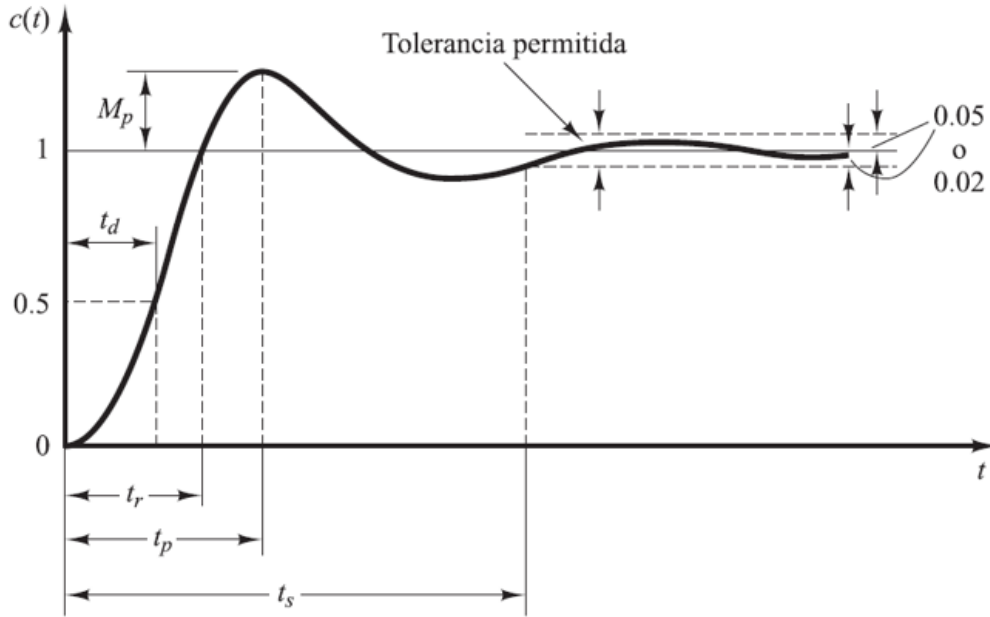


Figura 6: Gráfica temporal de la respuesta de un sistema de segundo orden subamortiguado, tomado de [20]

5.4.2. Tiempo pico

El tiempo de pico t_p se obtiene derivando la ecuación temporal (4) y evaluando la respuesta en $t = t_p$, de esa forma podemos encontrar el tiempo de máximo pico:

$$t_p = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}} \quad (12)$$

5.4.3. Tiempo de establecimiento:

El tiempo de establecimiento viene dado por medio de una tolerancia permitida. Dicha tolerancia permitida puede ser del 2% o del 5%. Cuando el sistema oscila dentro de esa tolerancia podemos decir que el sistema de segundo orden se encuentra dentro del régimen permanente.

$$2\% \rightarrow t_s = \frac{4}{\zeta \omega_n} \quad (13)$$

$$5\% \rightarrow t_s = \frac{3}{\zeta \omega_n} \quad (14)$$

5.4.4. Máximo sobreimpulso

Es usado para medir cuánto la señal sobrepasa la referencia con relación a su estado estacionario. El máximo sobrepaso o sobreimpulso (overshoot) puede medirse:

$$M_p = 100e^{\frac{-\pi \zeta}{\sqrt{1-\zeta^2}}} \quad (15)$$

5.4.5. Método de fracciones parciales para obtener la respuesta temporal de un sistema

Abordando los diferentes casos de fracciones parciales, se puede resumir casos particulares cuando la función de Laplace tiene raíces reales distintas, raíces reales repetidas o raíces complejas conjugadas, para cada caso se tiene que:

1. Polos reales simples

Cuando la función de Laplace tiene raíces reales distintas en el denominador (es decir, polos reales simples), la descomposición toma la forma:

$$\frac{N(s)}{(s-p_1)(s-p_2)} = \frac{A}{s-p_1} + \frac{B}{s-p_2} \quad (16)$$

Siendo N(s) el numerador de la función, se resuelve para las constantes A, B usando igualando coeficientes con la función de Laplace original o evaluando en los polos.

Luego se aplica la transformada inversa de Laplace:

$$\mathcal{L}^{-1} \left\{ \frac{A}{(s-p)} \right\} = A \cdot e^{p \cdot t} \quad (17)$$

2. Polos reales múltiples

Si hay un polo real repetido de orden n, por ejemplo en $(s - p)^n$, se expande así:

$$\frac{N(s)}{(s-p)^n} = \frac{A_1}{s-p} + \frac{A_2}{(s-p)^2} + \dots + \frac{A_n}{(s-p)^n} \quad (18)$$

Su transformada inversa de Laplace queda de la forma:

$$\mathcal{L}^{-1} \left\{ \frac{A}{(s-p)^n} \right\} = \frac{t^{n-1}}{(n-1)!} \cdot e^{p \cdot t} \quad (19)$$

3. Polos complejos conjugados

Si aparecen raíces complejas conjugadas del tipo $s = \alpha \pm j\omega$, el denominador contiene un

factor cuadrático irreducible:

$$\frac{N(s)}{(s^2 + 2\zeta\omega_n s + \omega_n^2)} = \frac{A s + B}{(s^2 + 2\zeta\omega_n s + \omega_n^2)} \quad (20)$$

Su transformada inversa de Laplace queda de la forma:

$$\mathcal{L}^{-1} \left\{ \frac{A s + B}{(s^2 + 2\zeta\omega_n s + \omega_n^2)} \right\} = e^{-\zeta\omega_n t} (C_1 \cos(\omega_d t) + C_2 \sin(\omega_d t)) \quad (21)$$

Esta descomposición permite escribir la respuesta temporal $H(t)$ como una suma de exponentes y senos, cosenos amortiguados, reflejando el comportamiento dinámico del sistema en el tiempo.

5.5. Polos de los sistemas de segundo orden

Partiendo de la ecuación general (1) de un sistema de segundo orden, los polos del sistema están dados por la ecuación característica (2), la cual se puede despejar y se obtiene:

$$s_{1,2} = \frac{-2\zeta\omega_n \pm \sqrt{4\omega_n^2(\zeta^2 - 1)}}{2} \quad (22)$$

Simplificando la ecuación (9) se obtiene la expresión para encontrar los polos del sistema de segundo orden:

$$s_{1,2} = -\zeta\omega_n \pm \omega_n \sqrt{(\zeta^2 - 1)} \quad (23)$$

Un sistema subamortiguado es aquel que posee un par de polos complejos conjugados dentro de un sistema de segundo orden. Analizando el sistema ante una entrada escalón y su ecuación (10), Cuando $0 < \zeta < 1$:

$$s_{1,2} = -\zeta\omega_n \pm j\omega_n \sqrt{(\zeta^2 - 1)} \quad (24)$$

Podemos llamar el segundo miembro de la ecuación anterior cómo la frecuencia natural amortiguada del sistema de segundo orden subamortiguado:

$$\omega_d = \omega_n \sqrt{(\zeta^2 - 1)} \quad (25)$$

Así, la ecuación (11) de los polos del sistema se puede resumir cómo:

$$s_{1,2} = -\zeta\omega_n \pm j\omega_d \quad (26)$$

Por ende, el diagrama de polos y ceros viene dado por:

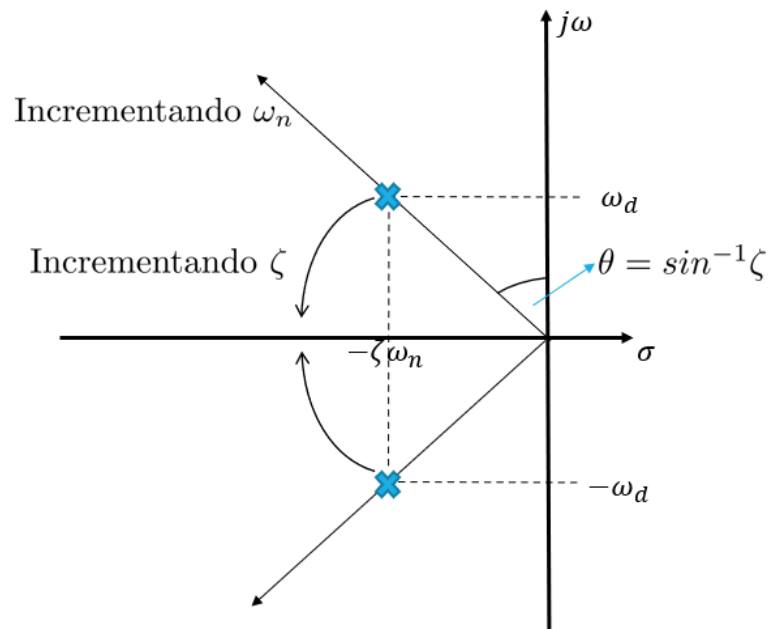


Figura 7: Diagrama de polos y ceros de un sistema subamortiguado de segundo orden, tomado de [20]

En la figura 7, se observa que si se incrementa ω_n los polos tienden a crecer radialmente. Por otro lado, si se aumenta ζ los polos tienden a acercarse al eje real, y si se disminuye tendiendo hacia cero, los polos se aproximan al eje imaginario.

5.6. Acción de control PID

El control PID es un mecanismo de control en lazo cerrado ampliamente utilizado en la automatización industrial y sistemas de control. Este controlador es esencial en aplicaciones donde se requiere mantener una variable de proceso (como la temperatura, presión, nivel o velocidad) en un valor deseado, compensando cualquier desviación de este valor causada por perturbaciones externas. Además, surge como consecuencia de la combinación de tres acciones básicas de control, las acciones proporcional, integral y derivativa. Estas tres acciones trabajan en conjunto para proporcionar un control preciso y estable, permitiendo que el sistema regrese al punto de consigna (setpoint) de manera eficiente, sin oscilaciones excesivas ni retardo significativo [20].

Sea un sistema con realimentación unitaria donde su proceso o planta es $G(s)$, con entrada $r(t)$, salida $y(t)$, perturbaciones $p(t)$ y cuyo controlador es $C(s)$, su diagrama de bloques se puede representar cómo la figura 8:

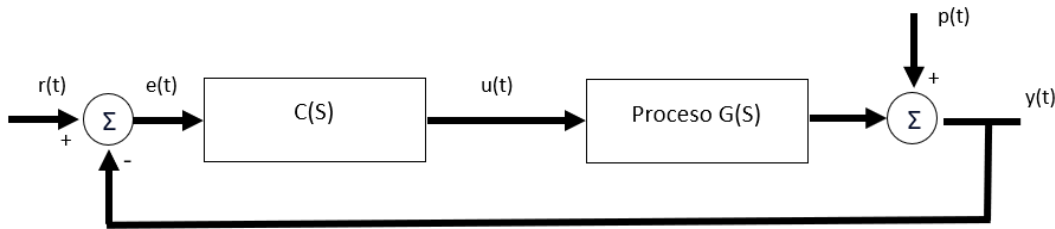


Figura 8: Diagrama de bloques de lazo cerrado con acción de control $C(s)$, entrada $r(t)$, perturbaciones $p(t)$, planta $G(s)$ y salida $y(t)$, tomado de fuente propia

El controlador $C(s)$ se puede descomponer en 3 acciones mencionadas anteriormente, las cuales están representadas en la siguiente figura 9, donde se incluyen cada una de ellas.

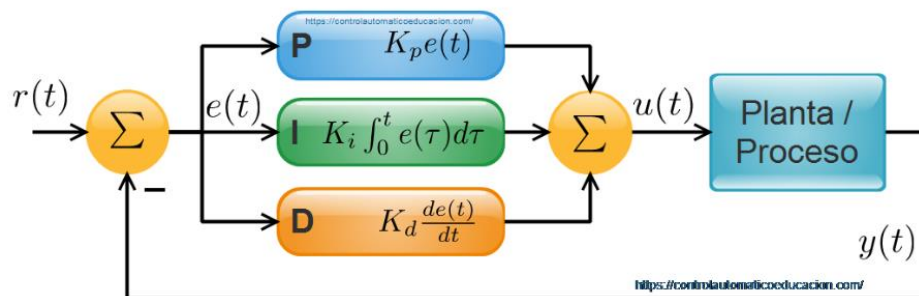


Figura 9: Diagrama de bloques de un sistema de lazo cerrado con su acción de control proporcional, acción integral y derivativa, tomado de [20]

La acción proporcional responde a la magnitud del error actual, la acción integral responde al error acumulado en el tiempo, y la acción derivativa responde a la tasa de cambio del error. Es importante destacar que el controlador PID siempre recibe a su entrada el error $e(t)$, por lo tanto estos tres componentes del controlador actúa sobre esta variable de entrada.

5.6.1. Acción proporcional (P)

La acción proporcional es el componente más básico del controlador PID. Esta acción produce una salida que es directamente proporcional al error del sistema, es decir, la diferencia entre el valor deseado (setpoint) y el valor medido del proceso.

En la Figura 10 se muestra el diagrama de bloques correspondiente a la acción proporcional (P) de un controlador. Esta acción genera una señal de control $u(t)$ que es proporcional al error $e(t)$, el cual resulta de la diferencia entre la señal de referencia $r(t)$ y la salida del sistema $y(t)$, mediante la ganancia proporcional

En este esquema, la señal denominada **Bias** representa un término constante que se puede utilizar para ajustar manualmente el valor de salida del controlador cuando el error es cero, permitiendo así corregir posibles desplazamientos o condiciones iniciales distintas de cero. Por otro lado, la señal marcada como **Q** representa una perturbación que afecta al proceso, lo que permite modelar condiciones externas o no deseadas que influyen sobre el comportamiento del sistema.

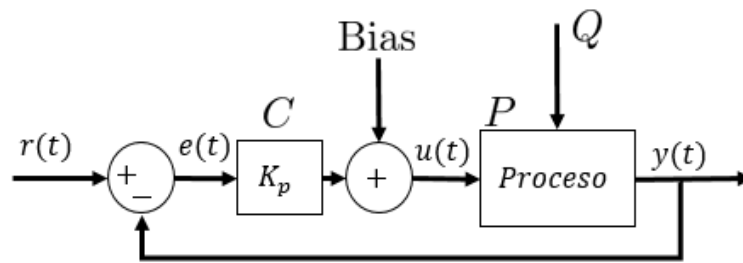


Figura 10: Diagrama de bloques, acción proporcional K_p , tomado de [20]

La ecuación que define la acción proporcional es:

$$P(t) = k_p \cdot e(t) \quad (27)$$

Donde:

- $P(t)$ es la salida proporcional en el tiempo t
- k_p es la ganancia proporcional, parámetro que determina la magnitud de la respuesta
- $e(t)$ es el error en el tiempo t , definido cómo $e(t) = r(t) - y(t)$, donde $r(t)$ es el setpoint

La ganancia proporcional K_p es un parámetro crítico que determina cuánto influye el error actual en la salida del controlador. Un valor alto de K_p hará que el sistema responda de manera más agresiva a los cambios en el error, lo que puede llevar a una rápida corrección del mismo. Sin embargo, si el K_p es demasiado alto, puede provocar oscilaciones o incluso inestabilidad, ya que el sistema podría sobrepasar repetidamente el setpoint.

Por otro lado, un valor bajo de K_p resultará en una respuesta más lenta y menos agresiva, lo que puede ser beneficioso para sistemas que requieren estabilidad, pero puede no ser adecuado si se necesita una rápida corrección del error.

5.6.2. Acción integral (I)

La acción integral se encarga de corregir el error acumulado en el tiempo, eliminando el offset residual que queda cuando solo se utiliza la acción proporcional. La acción integral considera el área bajo la curva del error en función del tiempo, sumando los errores pasados para ajustar la salida del controlador. El diagrama de bloques de un proceso con acción proporcional e integral se observa en la figura 11.

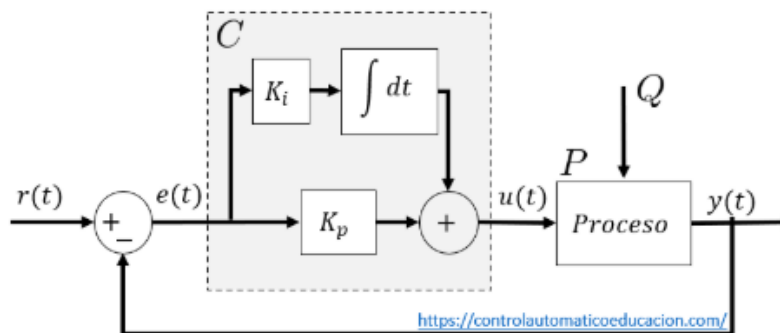


Figura 11: Diagrama de bloques de controlador con acción proporcional integral, tomado de [20]

La ecuación que define la acción integral es:

$$I(t) = k_i \cdot \int_0^t e(\tau) d\tau \quad (28)$$

Donde:

- $I(t)$ es la salida integral en el tiempo t
- k_i es la ganancia integral, que determina la influencia del error acumulado en la salida del controlador
- τ es una variable de integración que representa el tiempo pasado

El parámetro K_i ajusta la velocidad con la que el error acumulado influye en la salida del controlador. Una ganancia integral alta K_i resultará en una corrección rápida del error acumulado, lo que puede ser útil para sistemas donde es crucial eliminar el offset rápidamente.

Sin embargo, una ganancia integral demasiado alta puede llevar a oscilaciones y a un fenómeno conocido como integral windup (acumulación excesiva de la acción integral), donde el controlador continúa integrando el error incluso cuando el sistema ya ha alcanzado el setpoint, lo que puede

provocar una sobrecorrección y posterior oscilación [20].

5.6.3. Acción derivativa (D)

La acción derivativa predice el comportamiento futuro del error al calcular su tasa de cambio. Esta acción proporciona una corrección basada en la rapidez con la que el error está cambiando, y su principal función es mejorar la estabilidad del sistema y reducir la magnitud de las oscilaciones. El diagrama de bloques de un controlador tipo PID, el cual contiene las 3 acciones proporcional, integral, derivativa se representa cómo la figura 12.

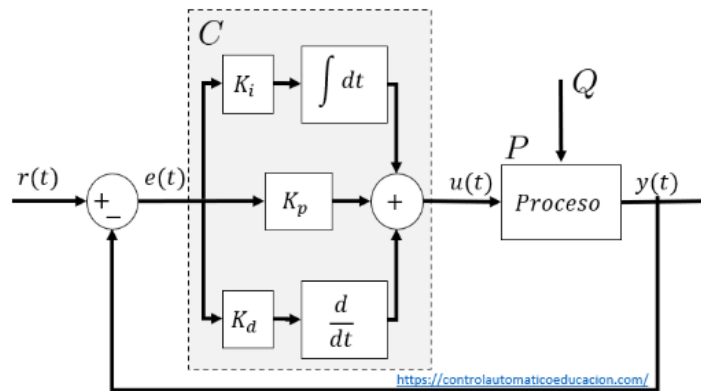


Figura 12: Diagrama de bloques controlador PID proporcional, integral, derivativo, tomado de [20]

La ecuación que define la acción derivativa es:

$$D(t) = k_d \cdot \frac{de(t)}{dt} \quad (29)$$

Donde:

- $D(t)$ es la salida del tiempo derivativo
- k_d es la ganancia derivativa, que determina la influencia de la tasa de cambio del error en la salida del controlador
- $\frac{de(t)}{dt}$ es la derivada el error respecto al tiempo

La ganancia derivativa K_d ajusta la sensibilidad del controlador a la tasa de cambio del error. Un valor alto de K_d puede ayudar a amortiguar las oscilaciones y mejorar la respuesta transitoria del sistema, haciendo que la salida del controlador reaccione más rápidamente a los cambios rápidos en el error.

Sin embargo, como la acción derivativa es muy sensible al ruido, un valor muy alto de K_d puede amplificar el ruido presente en la señal de error, lo que puede causar una respuesta errática del sistema. Por esta razón, en la práctica, a menudo se combina la acción derivativa con un filtro pasabajos para suavizar la señal y reducir la sensibilidad al ruido.

5.6.4. Combinación de las acciones del control PID

La verdadera potencia del controlador PID radica en la combinación de sus tres componentes. Cada una de las acciones – Proporcional, Integral, y Derivativa – contribuye de manera única al control del sistema. La acción proporcional proporciona la corrección necesaria basada en el error actual, la acción integral corrige errores pasados acumulados, y la acción derivativa prevé y ajusta posibles errores futuros.

La salida total del controlador PID se puede expresar como la suma de las tres acciones:

$$u(t) = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau) d\tau + k_d \cdot \frac{de(t)}{dt} \quad (30)$$

5.7. Diseño por asignación de polos

Para diseñar a través de este método, se debe conocer a priori el modelo matemático que representa la dinámica de la planta (función de transferencia). Para posteriormente poder manipular la ubicación de esos polos en el lazo cerrado de control y poderlos ubicar en el lugar deseado, obteniendo una respuesta especificada. Teniendo un controlador PID cómo se muestra en la figura 13:

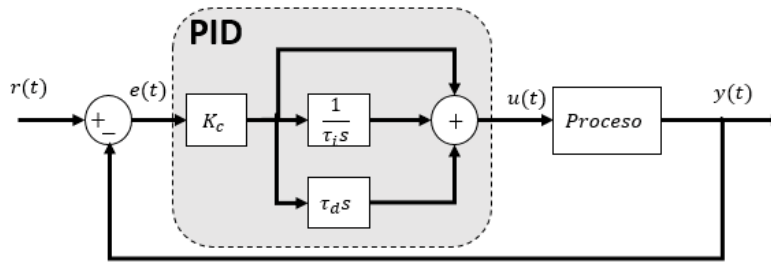


Figura 13: Diagrama de bloques de un proceso con controlador PID convencional, tomado de [20]

Dónde se puede pasar de tener el controlador PID de la figura 12 en términos de ganancias, a su forma convencional en términos de tiempo integral y tiempo derivativo (figura 13), teniendo en cuenta que:

$$k_p = k_c \quad (31)$$

$$k_i = \frac{k_c}{\tau_i} \quad (32)$$

$$k_d = k_c \cdot \tau_d \quad (33)$$

τ_i (tiempo integral) y τ_d (tiempo derivativo) tienen unidades de tiempo, lo que ayuda a entender cuánto tarda la acción integral en acumular un error o cuán rápido responde la acción derivativa.

Del diagrama de bloques de la figura 13, se puede obtener la ecuación del controlador, de manera que se puede ver y entender el funcionamiento de sus 3 parcelas:

$$C(s) = k_c \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) \quad (34)$$

Donde k_c es la ganancia proporcional, τ_i el tiempo integral del controlador y τ_d el tiempo derivativo del controlador. Multiplicando y dividiendo por $\tau_i s$ se puede reescribir la ecuación (34) cómo:

$$C(s) = \frac{k_c \tau_i \tau_d s^2 + k_c \tau_i s + k_c}{\tau_i s} \quad (35)$$

Sin embargo, la ecuación del control PID convencional es una función de transferencia no propia, lo que quiere decir que es una función con más ceros que polos. Y por lo tanto el controlador no es implementable en la vida real. Para solucionar ese problema, se adiciona un **filtro pasa-bajos** a la ecuación del PID en la acción derivativa al momento de implementar el controlador con el objetivo de volver la ecuación implementable y también para poder filtrar los ruidos en alta frecuencia que son amplificadas por el término derivador del controlador [21]. Así el control PID con el filtro pasa bajos viene dado cómo se muestra en la figura 14:

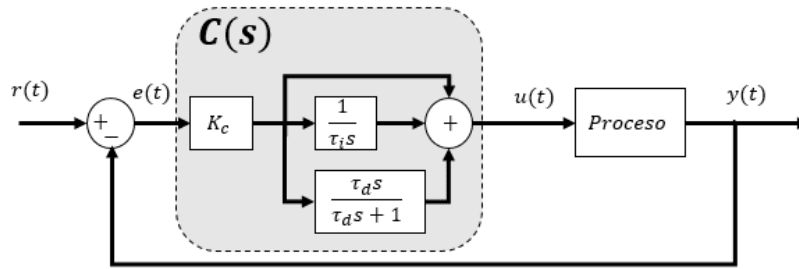


Figura 14 : Diagrama de bloques de un proceso con controlador PID y filtro pasa bajos, tomado de [20]

y por ende su ecuación general resulta:

$$C(s) = k_c \left(1 + \frac{1}{\tau_i s} + \frac{\tau_d s}{\alpha \tau_d s + 1} \right) \quad (36)$$

donde el parámetro $\alpha \in (0,1)$ es un parámetro de sintonía relacionado a cuestiones de atenuación de ruido y robustez del controlador. Reescribiendo la ecuación se tiene:

$$C(s) = \frac{(\alpha k_c \tau_i \tau_d + k_c \tau_i \tau_d) s^2 + (k_c \tau_i + k_c \tau_d) s + k_c}{\alpha \tau_i \tau_d s^2 + \tau_i s} \quad (37)$$

Si se denomina cómo A el numerador de la función de transferencia del proceso (ecuación 1), B su denominador, D el numerador de la función de transferencia del controlador (ecuación 35) y E su denominador, se tiene:

$$G(s) = \frac{K \omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2} = \frac{k}{s^2 + a s + b} = \frac{A}{B} \quad (38)$$

$$C(s) = \frac{k_c \tau_d s^2 + k_c s + \frac{k_c}{\tau_i}}{s} = \frac{d_2 s^2 + d_1 s + d_0}{s} = \frac{D}{E} \quad (39)$$

Donde:

$$a = 2\zeta\omega_n ; b = \omega_n^2 ; k = K \omega_n^2$$

$$d_2 = k_c \tau_d s^2 ; \quad d_1 = k_c ; \quad d_0 = \frac{k_c}{\tau_i}$$

La función de transferencia de lazo cerrado correspondiente al diagrama de bloques de la figura 8 es:

$$H(s) = \frac{C(s) \cdot G(s)}{1 + C(s) \cdot G(s)} = \frac{DA}{EB + DA} \quad (40)$$

Por ende, reemplazando los polinomios $G(s)$ de la ecuación (38) y $C(s)$ de la ecuación (39), se obtiene la función de lazo cerrado $H(s)$ de tercer orden que representa nuestro sistema, de manera que:

$$H(s) = \frac{k (d_2 s^2 + d_1 s + d_0)}{s (s^2 + a s + b) + k (d_2 s^2 + d_1 s + d_0)} \quad (41)$$

$$H(s) = \frac{k (d_2 s^2 + d_1 s + d_0)}{s^3 + (a + k d_2) s^2 + (b + k d_1) s + k d_0} \quad (42)$$

Por último, en este método solo falta asignar los polos al controlador para que la función de transferencia $H(s)$ se comporte como se quiere. Para ello se define una ecuación característica dominante de segundo orden:

$$P_d(s) = s^2 + h_1 s + h_2 \quad (43)$$

Donde $P_d(s)$ es el polinomio o ecuación característica deseada, que contiene los dos polos dominantes que van a gobernar la dinámica del sistema. Ya que $H(s)$ es de tercer orden, se completa el polinomio deseado con un polo rápido p_1 , el cual debe estar bien alejado del eje imaginario. Por ende:

$$P_d(s) = (s^2 + h_1 s + h_2)(s + p_1) \quad (44)$$

Se realiza el producto de los polos y de esta manera se llega a la ecuación característica deseada:

$$P_d(s) = s^3 + \alpha_1 s^2 + \alpha_2 s + \alpha_3 \quad (45)$$

De esta manera, cómo tienen el mismo orden tanto el denominador de $H(s)$ y el polinomio $P_d(s)$, se pueden igualar sus coeficientes y obtener una expresión para la ganancia k_c , el tiempo integral τ_i y el tiempo derivativo τ_d . Por lo tanto:

$$a + k d_2 = \alpha_1 \quad (46)$$

$$b + k d_1 = \alpha_2 \quad (47)$$

$$k d_0 = \alpha_3 \quad (48)$$

Reemplazando los valores de d_2, d_1, d_0 se llega a las expresiones generales para los parámetros del controlador PID.

$$k_c = \frac{\alpha_2 - b}{k} \quad (49)$$

$$\tau_i = \frac{k k_c}{\alpha_3} \quad (50)$$

$$\tau_d = \frac{\alpha_1 - a}{k k_c} \quad (51)$$

5.8. Lugar de raíces del controlador PID

5.8.1. ¿Qué es el lugar geométrico de las raíces?

El Lugar Geométrico de las Raíces (LGR), también conocido como método de root locus, es una técnica visual utilizada en control automático para analizar cómo se desplazan las raíces de la ecuación característica de un sistema de control a lo largo del plano complejo en función de la variación de un parámetro específico, típicamente la ganancia del sistema.

Por ejemplo, suponga que se tiene un proceso que es un horno de temperatura representado por una función de transferencia de primer orden y que a ese proceso se le coloca un controlador PI en lazo cerrado (ver figura 15).

Inicialmente el sistema en lazo cerrado tendrá dos polos y un cero ubicados en una determinada posición. Pero si se aumenta la ganancia del controlador PI, los polos comenzarán a moverse haciendo que la respuesta del sistema de control sea mucho más rápida. En cuanto el cero se mantendrá estático.

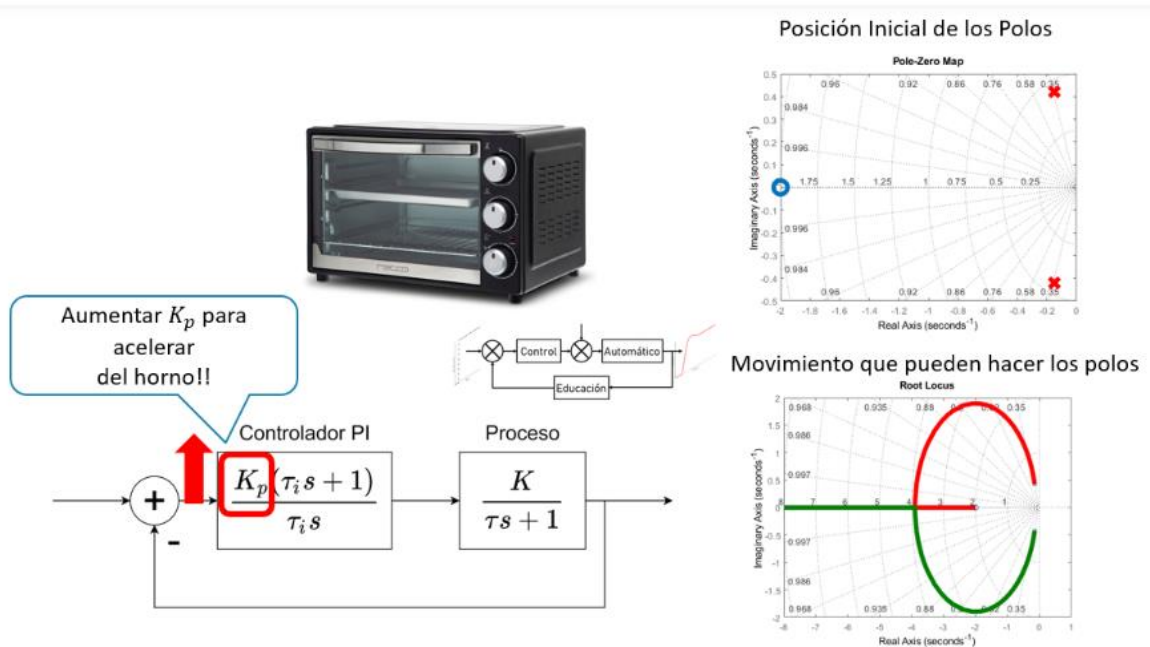


Figura 15: Lugar geométrico de las raíces y movimiento de los polos para el control de la temperatura de un horno con un controlador PI, tomado de [20]

En la figura 15 se puede ver el recorrido del polo superior en una traza roja, y el polo inferior en una traza verde en la medida que la ganancia del controlador se fue aumentando. Ese trazado muestra el lugar geométrico de las raíces, es decir el camino por donde los polos se desplazan conforme se modifica algún parámetro del sistema.

5.8.2. Método del lugar geométrico de las raíces

Para desarrollar este método, se debe partir inicialmente de un polinomio característico de variable compleja (s para tiempo continuo) y con coeficientes reales. Dicho polinomio dependerá de un parámetro real (k). Por ende:

$$P(c, k) = 0 \quad (52)$$

Donde: P es el polinomio, c es la variable compleja y k es el parámetro real que puede variar entre $(0, \infty)$ o en cualquier otro intervalo de interés.

El polinomio característico debe tener siempre **coeficientes reales**, porque esto va a permitir que el polinomio tenga simetría con sus raíces, donde dichas raíces podrán ser raíces complejas conjugadas. Esta suposición es válida desde que generalmente dichos polinomios característicos provienen de ecuaciones diferenciales de sistemas físicos reales donde dichos coeficientes van a depender de propiedades físicas como el área de un tanque, la abertura de una válvula, etc.

Cómo el polinomio característico depende del parámetro k, siempre será posible expresar dicho polinomio con la siguiente ecuación característica:

$$P(c, k) = A(c) + k B(c) \quad (53)$$

Donde se debe cumplir que:

- El polinomio A contenga los polos del sistema y el B los ceros del sistema
- El grado del polinomio A es mayor o igual al grado del polinomio B
- El polinomio A y B son mónicos (el coeficiente del máximo exponente será 1).
- La condición del módulo entre A(c) y B(c), sea:

$$\left| \frac{B(c)}{A(c)} \right| = \left| -\frac{1}{k} \right| = \frac{1}{k} \quad (54)$$

- La condición de la fase entre los polinomios sea:

$$\varphi\left(\frac{B(c)}{A(c)}\right) = \pm 180^\circ \pm 360p \quad \text{si } k > 0 \quad (55)$$

$$\varphi\left(\frac{B(c)}{A(c)}\right) = \pm 0^\circ \pm 360p \quad \text{si } k < 0 \quad (56)$$

Por ende, para determinar el ángulo o la fase sólo se necesita saber si k es positivo o negativo.

Los polinomios $A(c)$ y $B(c)$ se pueden expresar en forma polinómica cómo:

$$B(c) = (s - z_1)(s - z_2) \dots (s - z_m) \quad (57)$$

$$A(c) = (s - p_1)(s - p_2) \dots (s - p_n) \quad (58)$$

Dónde m y n son los órdenes de cada polinomio

De las propiedades de la fase de un cociente, se tiene que:

$$\varphi\left(\frac{X}{Y}\right) = \varphi(X) - \varphi(Y) \quad (59)$$

y la fase de un producto es:

$$\varphi(X.Y) = \varphi(X) + \varphi(Y) \quad (60)$$

aplicando las dos propiedades anteriores a la condición de fase, tenemos que:

$$\varphi(B(c)) - \varphi(A(c)) = \pm 180^\circ \pm 360p \quad (61)$$

$$\pm 180^\circ \pm 360p = \varphi(s - z_1) + \varphi(s - z_2) + \dots + \varphi(s - z_m) - \varphi(s - p_1) - \varphi(s - p_2) - \dots - \varphi(s - p_n)$$

Lo que convierte un problema de teoría de control en un problema de fase simple de resolver. Dónde, gráficamente se puede representar y ubicar los polos en el plano complejo S y aplicar la condición de ángulo o condición de fase para saber si un determinado punto se encuentra en el camino que recorre los polos conforme se aumenta la ganancia del sistema

5.8.3. Reglas de construcción del lugar geométrico de las raíces

Puntos especiales del LGR:

Partiendo de la ecuación 53 polinomial de análisis, existen algunos puntos especiales que serían el inicio y el fin del camino, en este caso considerando sólo el intervalo empleando la ganancia positiva para simplificar el análisis. Donde el grado del polinomio A es n ; el grado del polinomio B es m .

El polinomio A contiene los polos del sistema, estos comenzarán a moverse en la medida que se incrementa la ganancia de lazo cerrado y eventualmente con una ganancia infinita estos polos van a finalizar en los ceros contenidos en el polinomio B , o en infinito, si el polinomio B no tiene ceros suficientes

- Inicio del LGR: El LGR para control siempre inicia desde los polos cuando se

considera una ganancia baja o nula, $k=0$:

$$A(c) = 0 \quad (n \text{ raíces}) \quad (62)$$

- El fin del camino del LGR control siempre se dá cuando se tiene una ganancia exageradamente alta que tienda a infinito:

$$\frac{B(c)}{A(c)} = -\frac{1}{k} \quad (63)$$

$$\frac{B(c)}{A(c)} = -\frac{1}{\infty} \quad (64)$$

$$\frac{B(c)}{A(c)} = 0 \quad (65)$$

Si $n > m$ entonces $n-m$ raíces se van al infinito

Ramas del LGR:

Partiendo de que se tiene un sistema de lazo cerrado de control, donde se está variando un parámetro, en este caso la ganancia del sistema de control, se observa como los polos comenzarán a moverse hacia los ceros o hacia el infinito, trazando un camino (rama) por donde ellos han pasado, cómo se observa en la figura 16.

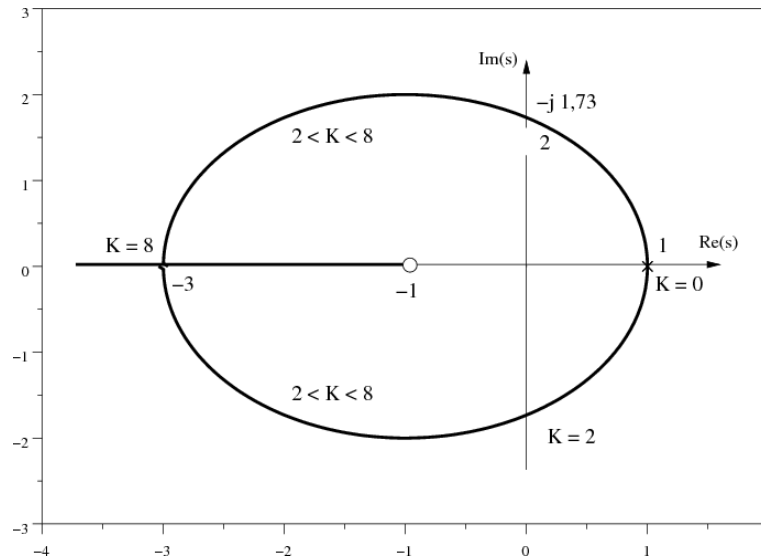


Figura 16: Ramas del LGR por donde se ubican los polos del sistema conforme se aumenta la ganancia proporcional, tomado de [20]

El número de ramas que se deben graficar siempre está relacionado con el grado del

polinomio A, si hay 3 polos, se deben graficar 3 ramas y si hay 6 polos, se deben graficar 6 ramas.

Simetría con el eje real del LGR en control:

El polinomio característico siempre posee coeficientes reales porque provienen de ecuaciones diferenciales de sistemas físicos reales donde dichos coeficientes van a depender de propiedades físicas. Esto permite que el polinomio tenga simetría con el eje real para todas sus raíces complejas conjugadas, por lo tanto:

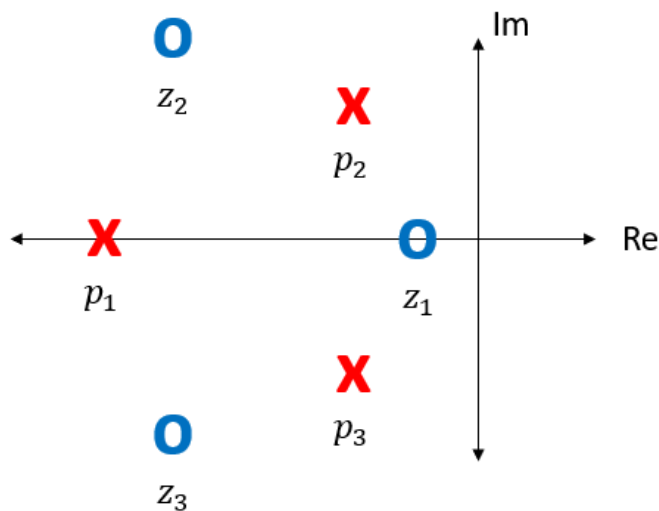


Figura 17: Simetría de los polos con respecto al eje real, tomado de [20]

Si una raíz es una solución del problema, su conjugado también tiene que ser solución del problema y esto es debido a que los polinomios trabajados poseen coeficientes reales, tal cómo se muestra en la figura 17.

Comportamiento asintótico del LGR:

En el análisis de sistemas de control mediante el LGR, nos encontramos con situaciones donde algunos polos no están dirigidos hacia ceros finitos del sistema. En estos casos, los polos se dirigen hacia el infinito en el plano complejo, y para comprender esta tendencia utilizamos lo que conocemos como comportamiento asintótico

Las asíntotas son líneas que describen la trayectoria de los polos a medida que se dirigen hacia el infinito. Se presentan cuando existe una diferencia en el número de polos y ceros; es decir, cuando el grado del polinomio característico del denominador (polinomios A) es mayor que el grado del polinomio del numerador (polinomios B), matemáticamente expresado como $n > m$.

Para determinar las asíntotas del LGR, se siguen ciertas reglas matemáticas y gráficas:

- **Calculando la ubicación de las asíntotas:** La dirección de las asíntotas se da por el ángulo que forman con el eje real positivo y se calcula con la fórmula:

$$\theta_{asintota} = \frac{(2k+1) 180^\circ}{n-m}, \quad k = 0, 1, 2, \dots, n-m-1 \quad (66)$$

Aquí, k representa el índice de la asíntota, $(2k + 1)$ representa los factores impares en la ecuación, n el número de polos y m el número de ceros.

- **Determinando el punto de salida hacia el infinito:** Los polos tienden a moverse hacia el infinito en direcciones que satisfacen las condiciones del sistema. Al no tener un cero atractor, su comportamiento asintótico es una parte crucial para entender la estabilidad y respuesta del sistema a medida que la ganancia aumenta.

Cruce en el eje real de las asíntotas:

Esta regla es aplicable únicamente cuando existen asíntotas en el sistema, lo cual ocurre si el número de polos (n) es mayor que el número de ceros (m). Se debe calcular el centroide σ de las asíntotas, como el promedio de la suma de las ubicaciones de los polos menos la suma de las ubicaciones de los ceros dividido entre la diferencia del número de polos y ceros, entonces:

$$\sigma_0 = \frac{\sum_{i=1}^n p_i - \sum_{i=1}^m z_i}{n-m} \quad (67)$$

Donde P_i representa la posición de los polos y z_i la de los ceros. El valor de σ_0 nos proporciona un punto de referencia en el eje real para determinar cómo se distribuirán las asíntotas en el plano s .

Por ejemplo, para el siguiente polinomio característico:

$$P = (s + 1)(s + 2)(s + 3) + k.1 \quad (68)$$

El centroide de las asíntotas es:

$$\sigma_0 = \frac{-1-2-3-0}{3-0} = -2 \quad (69)$$

Y su graficación de donde parten dichas asíntotas viene dado por la figura 18:

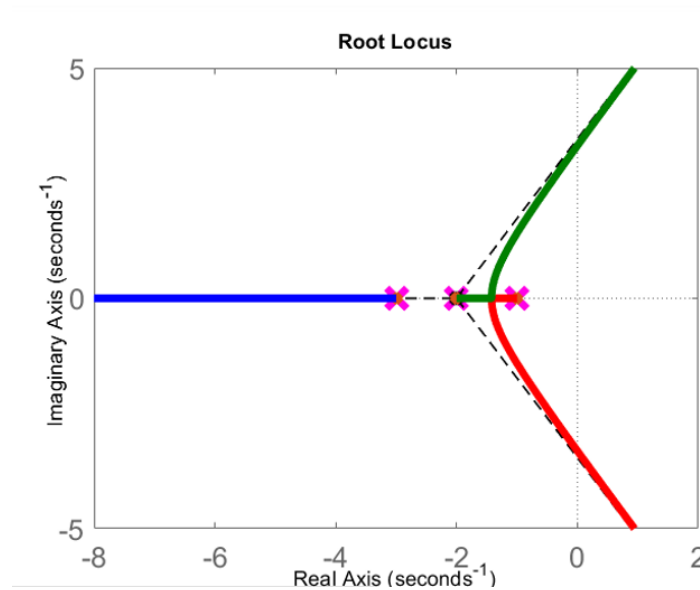


Figura 18: Asíntotas de las 3 ramas del lugar geométrico de las raíces, tomado de [20]

Lugar de las raíces en el eje real:

- Si las raíces del numerador o las raíces del denominador de la función de transferencia de lazo abierto se encuentran fuera del eje real, entonces no contribuirán con un cambio de fase en el eje real. Es decir, las raíces en el eje real tendrán una contribución de fase igual a cero, como se observa en la figura 19.
- Solamente se deben considerar las raíces en el eje real que ocasionen una contribución en la fase de 0 o de 180 grados. Segmentos en el eje real que dejen a la derecha un número impar de raíces (ya sean polos o ceros) tendrán una fase de $(180 \pm j360)$ para $(k > 0)$ y la fase complementaria para $(k < 0)$.

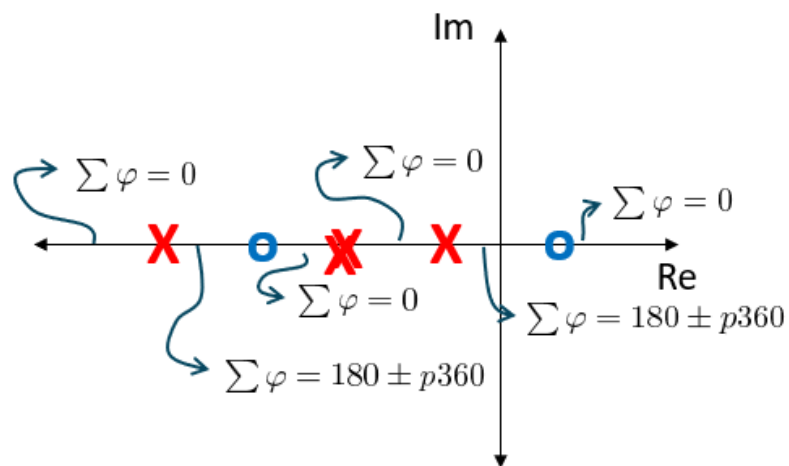


Figura 19: Polos y ceros con componente sólo en el eje real, tomado de [20]

Polos adyacentes en el eje real:

- Si existen dos polos adyacentes sobre el eje real, el lugar geométrico de las raíces abandona el eje, si existen dos ceros adyacentes sobre el eje real, el lugar geométrico de las raíces converge en el eje.
- Si un polinomio tiene una raíz múltiple en un punto, la derivada del polinomio también tendrá la raíz en ese mismo punto

Ángulo de partida en raíces complejas

Cuando se presentan raíces complejas conjugadas, es necesario calcular el ángulo de partida del polo o el ángulo de llegada al cero para saber la dirección de la rama.

- **Ángulo de partida del polo:**

$$\phi = 180^\circ - \sum \angle Polos + \sum \angle Ceros \quad (70)$$

- **Ángulo de llegada al cero:**

$$\beta = 180^\circ + \sum \angle Polos - \sum \angle Ceros \quad (71)$$

Cruce con el eje imaginario:

Los puntos en los cuales el lugar geométrico de las raíces corta al eje imaginario se determinan haciendo en la ecuación característica $c=j\omega$, igualando la parte real y la parte imaginaria a cero y resolviendo las ecuaciones resultantes para ω y k . Los valores obtenidos, dan la localización del punto de corte con el eje imaginario y la ganancia en dicho punto.

Ejemplo:

$$\begin{aligned} s^3 + 3s^2 + 2s + k &= 0 \\ (j\omega)^3 + 3(j\omega)^2 + 2(j\omega) + k &= 0 \\ k - 3\omega^2 + j(2\omega - \omega^3) &= 0 \\ k - 3\omega^2 &= 0 \\ 2\omega - \omega^3 &= 0 \end{aligned} \quad (72)$$

Solucionando el sistema de ecuaciones anterior se obtiene el cruce con el eje imaginario del lugar geométrico de las raíces

5.9. GeoGebra

GeoGebra es un software matemático dinámico para todos los niveles educativos que reúne geometría, álgebra, hojas de cálculo, gráficas, estadísticas y cálculo en un solo motor [5]. GeoGebra soporta dos lenguajes - GGBScript y Javascript.

5.9.1. Guiones GeoGebra Script

Cuando las opciones habituales de GeoGebra no resultan ágiles o adecuadas para los destinatarios o para plantear cierto escenario, puede resultar conveniente apelar a los guiones: los guiones de GeoGebra, GeoGebra Script, permite crear construcciones muy simples de usar y de muy alta interactividad. Como se desarrollará a continuación, hay modos muy sencillos de elaborar scripts o guiones con GeoGebra, que son programas breves.

Los siguientes son los empleos más habituales de los guiones:

- Control de respuestas ingresadas en un campo de texto.
- Creación de nuevas tareas / desafíos tras la pulsación de un botón o por generación aleatoria o establecida dentro de un conjunto previamente diseñado.
- Inicio de animaciones o de desenvolvimiento de nuevas secciones de la construcción al completarse una tarea.

5.9.2. Lenguajes de guiones

Coexisten en GeoGebra dos lenguajes de guion: GeoGebra Script y JavaScript. Esta introducción se centrará en GeoGebra Script que emplea la sintaxis de los comandos de GeoGebra y por lo tanto es más sencilla de aprender para quienes están habituados a trabajar con GeoGebra. Las diferencias entre el lenguaje Java de guiones es notoria dado que el JavaScript es mucho más versátil. Aún así, para la mayor parte de las aplicaciones, GeoGebra Script es suficientemente potente[22].

Los guiones son, básicamente, no más ni menos que una secuencia de comandos de GeoGebra ejecutados en una de estas dos situaciones:

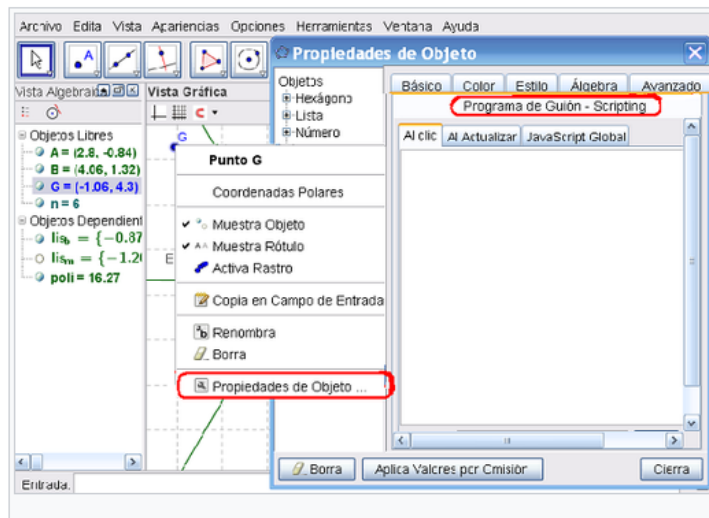
- **Clic:** tras un clic sobre un objeto.
- **Actualización:** cuando un valor o propiedad del objeto cambia. Esto puede suceder si se desplaza un elemento de la construcción, cambia el valor de un deslizador, etc.

Los guiones de GeoGebra están centrados en los objetos por lo que se puede crear uno por cada objeto sea tras un clic y/o desencadenado por su actualización. Solo los guiones de Java pueden eludir el objeto que les dé cabida.

Para añadir un guion a un objeto, es preciso dirigirse a la pestaña del **Cuadro de Propiedades Objeto**. Allí se exponen tres pestañas:

- **Al Clic**
- **Al Actualizar**
- **JavaScript Global**

Dejando de lado esta última pestaña, las dos primeras permiten vincular guiones al objeto seleccionado, cómo lo muestra la figura 20.



[22] **Figura 20: Interfaz de GeoGebra, pestañas para editar en GeoGebra Script y JavaScript, tomado de**

5.9.3. Botones

Los guiones desencadenados al clic sobre un botón son idénticos a los de clic sobre cualquier otro objeto pero los botones tienen ya una tradición como impulsores de un efecto y por eso parece más natural emplearlos en estos casos. Los botones se crean empleando la herramienta Tool Insert Button.gif Botón.

5.9.4. Cajas de entrada

Las cajas de entrada también se denominan campos de texto y se vinculan a la herramienta de casilla de entrada . Son probablemente los objetos más poderosos de GeoGebra. Permiten el ingreso de texto, números, funciones y cualquier otra entrada que se podría anotar manualmente, para convertirla de texto a objeto de GeoGebra.

Hay muchas maneras de usar las cajas de entrada en las construcciones: asociadas a objetos y a un guión al clic o procesándolas a posteriori usando otro objeto (por ejemplo, un botón) como se explicará a continuación.

5.9.5. Cajas de entrada vinculadas

Son cajas vinculadas a un objeto de GeoGebra existente: esto implica que exponen el valor del objeto que, a su vez, se modifica cuando se anota otro. Esto permite cambiar, por ejemplo, la definición de una función o cualquier valor numérico sin emplear deslizadores.

Básicamente, son cajas de entrada que pueden cambiar un simple objeto sin riesgos de cambios dramáticos en el resto de una construcción o de la hoja dinámica en general.

5.9.6. Cajas vinculadas en construcciones

El siguiente ejemplo ilustra cómo pueden crearse cajas vinculadas en una construcción, ver la figura 21:

- Abrir una ventana en blanco de GeoGebra
- Ingresar una función como, por ejemplo, $f(x) = x^2$, en la barra de entrada
- Seleccionar la herramienta Tool Insert Textfield.gif casilla de entrada (se encuentra en la penúltima caja de la barra).
- Un clic en cualquier lugar de la hoja dinámica desplegará una caja de diálogo en la que se puede ingresar " $f(x) =$ " y como Objeto Vinculado, seleccionar la función f .
- Al desplazar el gráfico, el campo de texto automáticamente expondrá la nueva definición de la función. También se puede modificar el contenido del campo de texto para ver cómo cambia la definición de la función y su gráfico.

Nota: Tal como se señaló, además de funciones, se puede modificar cualquier objeto que se desee,

sean valores numéricos o contenidos de texto.

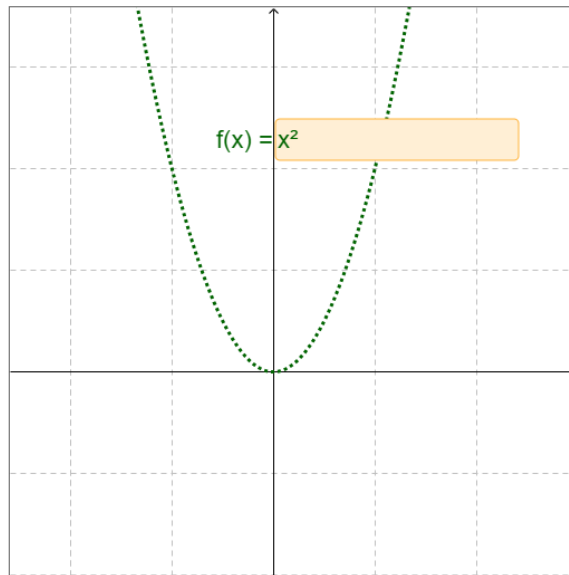


Figura 21: Ejemplo de la creación de cajas vinculadas en una construcción, tomado de [22]

5.9.7. JavaScript en GeoGebra

Los métodos más utilizados de JavaScript son Date, Number, Array y String. Además de la terminología de programación en inglés: output, statements, syntax, comments, variables, operators, arithmetic, assignment, data types, functions, strings, numbers, array, math, random, booleans, comparisons, conditions, switch, for, while, break, type conversion, constant, and debugging [22]

Pasos a seguir:

1. No escriba el código primero desde la ventana de GeoGebra JavaScript utilice Notepad + + para escribir el código y GeoGebra JavaScript para ejecutar el código.
2. Si el código tiene errores de bucle infinito o muchos datos para analizar con código incorrecto salen muchas ventanas emergentes con errores por cada dato en la lista y la aplicación puede cerrarse por sobrecarga y perder el código escrito en GeoGebra sino lo guardo después de haberle hecho modificaciones.
3. Pruebe los códigos con pocos elementos en las listas no más de tres elementos en la lista para evitar muchas ventanas emergentes si hay errores de código.
4. Aproveche las funcionalidades de un editor de código fuente gratuito en JavaScript como Notepad++ para que sea más fácil y cómodo el escribir y ver el código en JavaScript. Funcionalidades como autocompletado, reemplazar texto, clonar y mover vista, duplicar línea, atajos de teclado de Notepad++, etc.

5. Revise siempre que modifique el código en JavaScript que el modo Scripting está en la opción JavaScript y no GeoGebra Script porque sino va a tener errores de compilación y la palabra clave if de JavaScript cambia a If de GeoGebra Script causando otros errores de sintaxis.
6. El color de Programación JavaScript aparece en la ventana Scripting cuando cambia el modo de Scripting de GeoGebra Script a JavaScript y sales de las ventanas del mismo objeto y vuelves a entrar a la ventana de Scripting del objeto.
7. Siempre guarde primero el código escrito antes de ejecutarlo para evitar perder el código escrito en GeoGebra sino lo guardo por errores en el código.
8. GeoGebra no guarda automáticamente el código luego de hacerle modificaciones.

6. METODOLOGÍA

En esta sección se describe la metodología que se llevó a cabo para diseñar el controlador PID del sistema en GeoGebra. Primeramente, se realizó una conceptualización de los métodos para diseñar controladores PID. Entre los cuales, se encontró el método de diseño de controladores PID de Ziegler y Nichols, el método de diseño por asignación de polos y el método de diseño por el lugar geométrico de las raíces. El proyecto tiene el alcance para diseñar el controlador con dos métodos. Por lo cual, se descartó el método de Ziegler y Nichols, ya que con este método la respuesta del sistema tiende a ser agresiva y únicamente se puede utilizar si la dinámica del sistema es de tipo sigmoidal. Para este proyecto en específico se excitó el sistema con una entrada tipo escalón.

6.1. Método de asignación de polos:

Para el diseño y análisis de controladores PID aplicados a sistemas dinámicos, en este método de asignación de polos, se desarrolló una plataforma interactiva basada en tecnologías web y ecuaciones simbólicas. Esta herramienta permite al usuario ingresar los parámetros de diseño deseados como: el tiempo de asentamiento, sobreimpulso, tolerancia y el múltiplo del polo rápido junto con la función de transferencia del proceso $G(s)$ que el usuario desee. Adicionalmente, se puede seleccionar el tipo de controlador (P, PI, PD o PID) y ajustar el valor del filtro pasabajos α . Toda esta información se procesó mediante un entorno visual basado en GeoGebra, que brinda retroalimentación inmediata sobre el comportamiento del sistema y la evolución de su respuesta temporal.

La aplicación web está compuesta por dos partes fundamentales: el frontend y el backend. El frontend corresponde a la interfaz visual que el usuario observa en su navegador. Fue diseñado utilizando tecnologías estándar como HTML, CSS y JavaScript, y permite al usuario interactuar directamente con la plataforma mediante formularios, menús desplegables, botones y representaciones gráficas. Esta interfaz se encarga de mostrar tanto la función de transferencia, como la ubicación de los polos del sistema y su respuesta ante una entrada tipo escalón, integrando dinámicamente la herramienta GeoGebra para una visualización interactiva. Por otro lado, el backend representa la parte lógica y de procesamiento del sistema, encargada de realizar los cálculos matemáticos complejos. Fue desarrollado utilizando Flask, un microframework en Python que permite construir servidores web de forma sencilla. Flask actúa como el “cerebro” del sistema: recibe solicitudes desde la interfaz web, procesa los datos (por ejemplo, calcula la transformada inversa de Laplace de $H(s)/s$), y devuelve el resultado listo para ser graficado. La figura 22-1 muestra la arquitectura general del método propuesto. En la cual, se observa la interacción usuario-interfaz. Primeramente el usuario se encarga de ingresar los parámetros de diseño que requiere en la página web, la página web almacena los datos mediante código Javascript. Al tiempo Geogebra se encuentra embebida en esta página. Los datos son procesados y desde Javascript (Frontend junto con el HTML) se manda la solicitud al backend para hacer los cálculos de la inversa de Laplace. Luego se retornan al frontend y por último son graficados en Geogebra, el cual interacciona con el usuario a través de la página web.

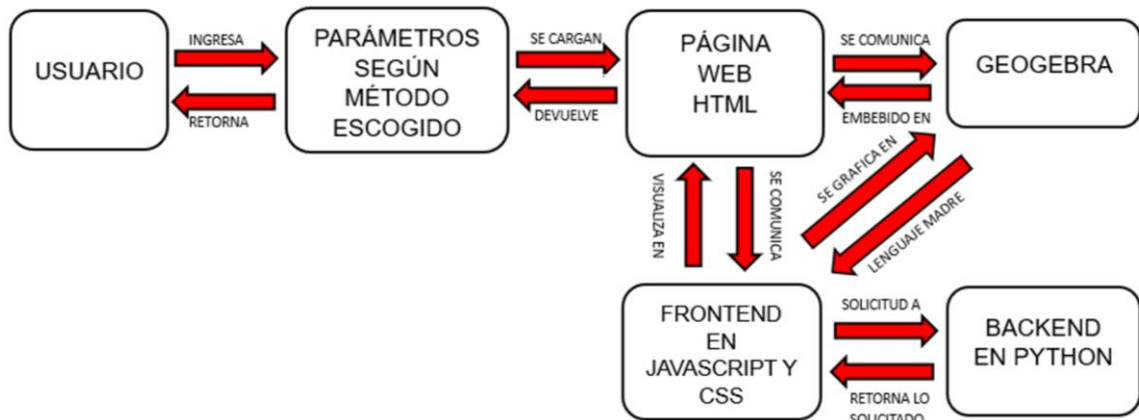


Figura 22-1: Diagrama de la arquitectura del método propuesto.

En la figura 22-2, se muestra el frontend creado para este método, utilizando cómo ejemplo una función de transferencia $10/(s^2 + 2s + 10)$ sin controlador y los parámetros de diseño Tss, Mp, Tolerancia, múltiplo del polo rápido y filtro pasabajos para dar aspecto visual de esta aplicación.

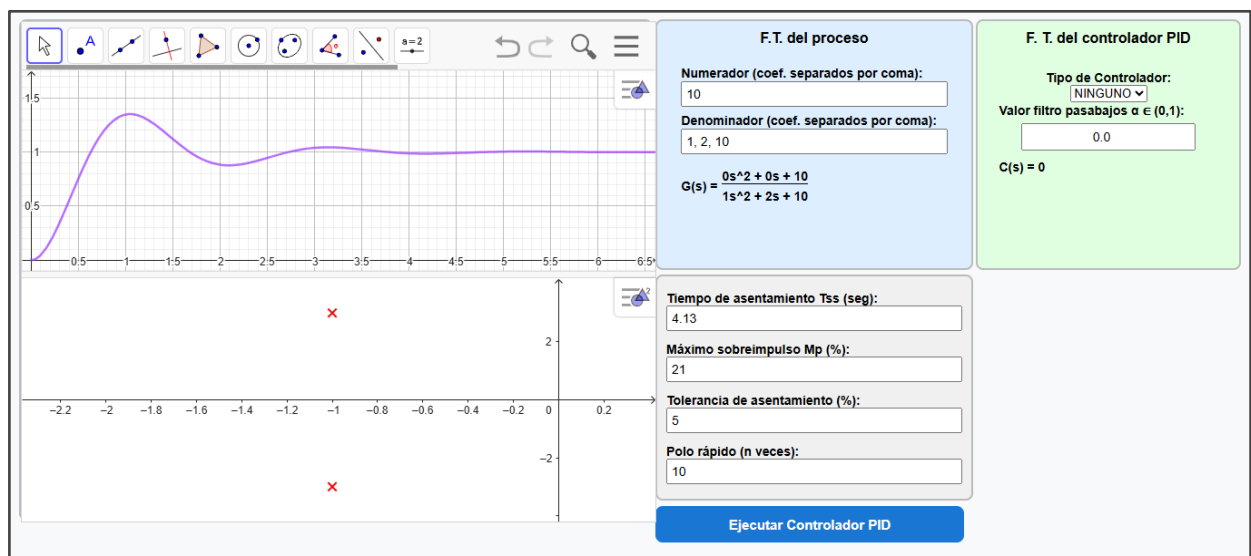


Figura 22-2: Interfaz del frontend con la respuesta temporal y polos del sistema mostrados en la vista gráfica de GeoGebra de la función de transferencia $10/(s^2 + 2s + 10)$

Para tener claro el funcionamiento de esta aplicación, se muestra el diagrama de flujo de caso de uso para el método de asignación de polos en la figura 23, el cual muestra los parámetros iniciales que debe ingresar el usuario, junto con el flujo y la lógica implementada para encontrar los

parámetros del controlador, los polos y la respuesta temporal requerida. En la figura 33, se encuentra el otro diagrama de caso de uso para el segundo método propuesto.

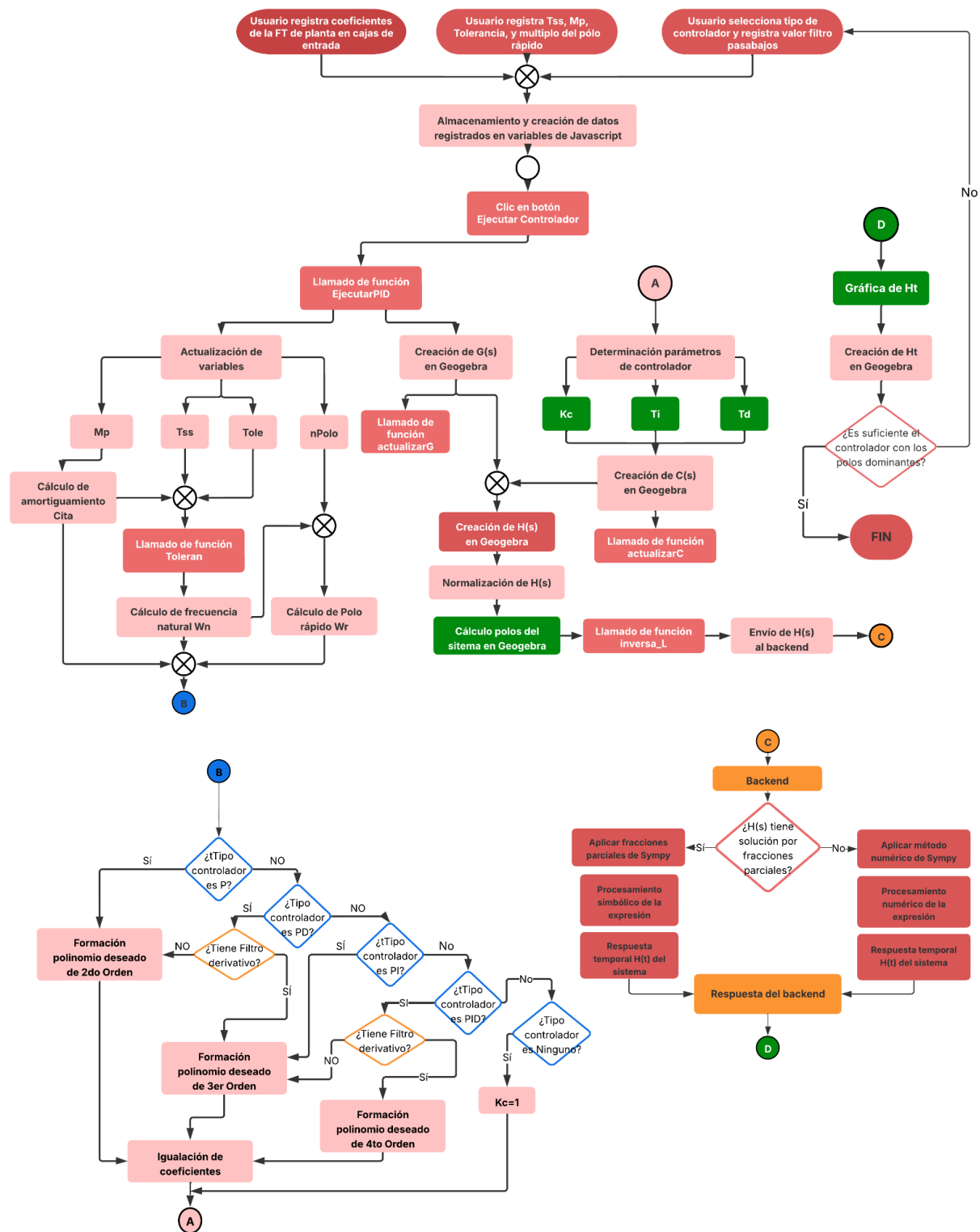


Figura 23: Diagrama de flujo para el caso de uso del método de asignación de polos.

A continuación, se explica detalladamente cada parte fundamental del frontend y backend de la aplicación web. Para este método se utilizó la siguiente función de transferencia de planta:

$$G(s) = \frac{2}{s^2 + 12s + 20,2} \quad (1)$$

La cual, se obtiene del control de la velocidad de un motor DC, planteada en el artículo “*DC Motor Speed: PID Controller Design*” [23]. Esto con el objetivo de comparar la confiabilidad de este método propuesto y la desarrollada en el artículo. Además, se trabajó con los mismos parámetros de diseño como tiempo de asentamiento, sobre impulso y tolerancia.

6.1.1 Procesamiento del frontend

Para el procesamiento de datos, se comenzó por guardar como string la función de transferencia que el usuario digita en la casilla de texto F.T del proceso de la interfaz creada. Además, se almacenó en variables los parámetros de diseño que también debe ingresar el usuario en las casillas correspondientes de la interfaz (figura 22), tales como: tiempo de establecimiento **Tss**, sobre impulso máximo **Mp**, tolerancia y múltiplo del polo rápido (línea 544-562 de la figura 24). Con el valor de sobre impulso se calculó la constante de amortiguamiento **Cita** a través de la [ecuación 15](#) del marco teórico, en la línea 568 de la figura 24. Seguidamente, con este valor, la tolerancia ingresada y el tiempo de establecimiento, se hizo el llamado a la función “**toleran()**” de javascript, la cual se encargó de calcular la frecuencia natural **wn**, mediante las [ecuaciones 13 y 14](#). Luego, en las líneas de código 576 hasta la 592 se guardaron en variables los coeficientes del numerador y denominador de la función G(s) de planta, dependiendo del grado de estos, así mismo se le asignaron las variables **a2**, **a1**, **a0** a los coeficientes de los términos s^2 , s^1 , s^0 en el numerador. Y en el mismo orden descendente las variables **b2**, **b1**, **b0** a los coeficientes del denominador. Por último, se armó la función G(s) en GeoGebra con estos coeficientes y mediante el comando **ggbApplet.evalCommand()**, que permite enviar instrucciones desde JavaScript hacia la instancia embebida (applet) de GeoGebra, en cuyo caso el resultado es el mismo si el usuario digitara en la vista algebraica de GeoGebra algún comando en específico. Por ende, se creó en GeoGebra la función G(s) como se observa en la línea 594 de la figura 24.

```

538 //limpia el entorno de Geogebra
539 const allObjects = ggbApplet.getAllObjectNames();
540 for (let i = 0; i < allObjects.length; i++) {
541   const objName = allObjects[i];
542   ggbApplet.deleteObject(objName);
543 }
544 //Actualizar valores de Tss, Mp y Tole
545 const TssValor = parseFloat(TssInput.value);
546 const MpValor = parseFloat(MpInput.value);
547 const ToleValor = parseFloat(ToleInput.value);
548 const nVecesP = parseInt(PoloRapidoInput.value); //Multiplo del polo rapido
549 const alphaPID = parseFloat(alphaInput.value);
550
551 if (!isNaN(TssValor)) Tss = TssValor;
552 if (!isNaN(MpValor)) Mp = MpValor;
553 if (!isNaN(ToleValor)) Tole = ToleValor;
554 if (!isNaN(nVecesP)) nPoloRapido = nVecesP;
555 if (!isNaN(alphaPID)) alpha = alphaPID;
556
557 console.log("---Valores actuales:---");
558 console.log("Tss:", Tss);
559 console.log("Mp:", Mp);
560 console.log("Tole:", Tole);
561 console.log("alpha:", alpha);
562 console.log("Multiplo Prapido:", nPoloRapido);
563
564 ggbApplet.setPerspective("G/D"); //Cambia Geogebra a vista 1 y 2 simultaneas
565
566 // Calcular el coeficiente de amortiguamiento
567 var Cita = -Math.log(Mp / 100) / Math.sqrt(Math.pow(Math.PI, 2) + Math.pow(Math.log(Mp / 100), 2));
568 Cita = parseFloat(Cita.toPrecision(5));
569 console.log(`cita= ${Cita}`);
570 // Calcular la frecuencia natural
571 var Wn = toleran(Tole, Cita, Tss);
572 Wn = parseFloat(Wn.toPrecision(4));
573 console.log("wn= ", Wn);
574
575 //Coeficientes de G(s)
576 if (!numeradorG || !denominadorG) {
577   console.error("Error: numeradorG o denominadorG no están definidos.");
578   return;
579 }
580
581 ordenNum = numeradorG.length - 1;
582 ordenDen = denominadorG.length - 1;
583 a2 = ordenNum >= 2 ? numeradorG[ordenNum - 2] : 0;
584 a1 = ordenNum >= 1 ? numeradorG[ordenNum - 1] : 0;
585 a0 = numeradorG[ordenNum];
586
587 b2 = ordenDen >= 2 ? denominadorG[ordenDen - 2] : 0;
588 b1 = ordenDen >= 1 ? denominadorG[ordenDen - 1] : 0;
589 b0 = denominadorG[ordenDen];
590 console.log("coeficientes numerador G= ", a2, a1, a0);
591 console.log("a0:", a0);
592 console.log("coeficientes denominador G= ", b2, b1, b0);
593
594 ggbApplet.evalCommand(`G(s) = (${a2}*s^2 + ${a1}*s + ${a0})/(${b2}*s^2 + ${b1}*s + ${b0})`);

```

Figura 24: Código en Javascript del frontend para determinar el coeficiente de amortiguamiento, la frecuencia natural y la función de planta G(s)

Continuando con el desarrollo del frontend, se procedió a guardar la selección del tipo de controlador que el usuario registró en el cuadro de texto F.T del controlador de la interfaz web, cómo se observa en la figura 25. Luego, se inicializaron las variables a utilizar cómo los parámetros del controlador K_c , t_i , t_d y los coeficientes del polinomio característico h_3 ,

h_2 , h_1 , h_0 . Cabe recordar que la finalidad de este método es aproximar la respuesta a un sistema de segundo orden. Por esa razón, dependiendo del controlador y el grado de la función digitada por el usuario, ya que el polinomio característico aumenta su grado según lo seleccionado, así mismo cambia las ecuaciones para calcular estos coeficientes. Por ende, se creó un condicional **if** dependiendo del tipo de controlador seleccionado y en cada condicional, se igualaron los coeficientes entre polinomio característico y coeficientes del denominador de la función de lazo cerrado. Con lo cual se obtuvo los parámetros k_c , t_i y t_d . Por último, se armó en GeoGebra la función de transferencia $C(s)$ del controlador utilizando **evalCommand**. También, se manejó el caso de no tener ningún controlador en el proceso, si el usuario así lo desea, como lo muestra la figura 26.

```

595 //-----CONTROLADOR PID-----
596 // Obtener Selección de tipo de controlador //'P','PI','PD','PID'
597 tipoControlador = document.getElementById("tipo-controlador").value;;
598 console.log("tipo Controlador:",tipoControlador);
599
600 // Reiniciar variables antes de calcular
601 kc = 0, ti = 0, td = 0;
602 let h0=0,h1=0,h2=0,h3=0, h4=0;
603
604 //Polo rápido del sistema Wr (nPoloRápido es el factor de seguridad)
605 let Wr=Wn*nPoloRapido;
606 console.log("Polo rapido Wr: ",Wr);
607
608 // Cálculo de parámetros según el tipo de controlador
609 if (tipoControlador === "P") {
610     //Coeficientes polinomio deseado de 2do orden  $s^2 + h_1s + h_0$ 
611     h3 = 0;
612     h2 = 1;
613     h1 = 2 * Cita * Wn;
614     h0 = Wn * Wn;
615
616     //Parámetros de diseño  $k_c$ 
617     kc = (h0 - b0) / (a0 !== 0 ? a0 : 1e-6);
618     ggbApplet.evalCommand(`C(s) = (${kc})`);
619
620 } else if (tipoControlador === "PI") {
621     //Coeficientes polinomio deseado de 3er orden  $s^3 + h_2s^2 + h_1s + h_0$ 
622     h3 = 1;
623     h2 = 2 * Cita * Wn + Wr
624     h1 = Wn**2 + 2*Cita*Wn*Wr
625     h0 = Wr*Wn**2
626
627     //Parámetros de diseño  $k_c, t_i$ 
628     kc = (h1 - b0) / (a0 !== 0 ? a0 : 1e-6);
629     ti = kc*a0 / h0;
630     ggbApplet.evalCommand(`C(s) = Expand(${kc} * (1 + 1/(${ti}*s)))`);
631
632 } else if (tipoControlador === "PD") {
633     if(alpha===0){
634         //Coeficientes polinomio deseado de 2do orden  $h_2s^2 + h_1s + h_0$ 
635         h3 = 0;
636         h2 = 1;
637         h1 = 2*Cita*Wn;
638         h0 = Wn**2 ;
639
640         //Parámetros de diseño  $k_c, t_d$ 
641         kc = (Wn**2 - b0)/a0 ;
642         td = (h1-b1)/(a0*kc) ;
643     }

```

Figura 25: a) Primera parte código en Javascript del cálculo de los parámetros k_c , t_i , t_d según el tipo de controlador

```

645 }else if(alpha >= 0.01 && alpha <= 0.1){
646
647     //Coeficientes polinomio deseado de 3er orden  $s^3 + h2*s^2 + h1*s + h0$ 
648     h3 = 1;
649     h2 = 2* Cita *Wn + Wr;
650     h1 = Wn**2 + 2*Cita*Wn*Wr;
651     h0 = Wr*Wn**2;
652     //Parámetros de diseño kc,td
653     td = 1/(alpha*(h2-b1)) ;
654     kc = (h1-b0-(b1/(alpha*td))) / (a0 +(a0/alpha));
655 }
656 ggbApplet.evalCommand(`C(s)=Expand(${kc}*(1+(${td}*s /(${alpha*td} *s +1))))`);
657
658 } else if (tipoControlador === "PID") {
659
660     if (alpha===0){
661         //Coeficientes polinomio deseado de 3er orden  $h3*s^3 + h2*s^2 + h1*s + h0$ 
662         h3 = 1;
663         h2 = 2*Cita*Wn + Wr
664         h1 = Wn**2 + 2*Cita*Wn*Wr
665         h0 = Wr*Wn**2
666
667         //Parámetros de diseño kc,ti,td
668         kc = h1-b0/a0
669         td = h2-b1/(a0*kc)
670         ti = a0*kc/h0
671
672     }else if (alpha >= 0.01 && alpha <= 0.1){
673
674         //Coeficientes polinomio deseado de 4to orden  $s^4+ h3*s^3 + h2*s^2 + h1*s + h0$ 
675         h3 = 2* Cita *Wn + (2*Wr);
676         h2 = Wn**2 + 4*Cita*Wn*Wr + Wr**2;
677         h1 = 2*Wr*Wn**2 + 4*Cita*Wn*Wr**2;
678         h0 = (Wn**2)*Wr**2;
679
680         //Parámetros de diseño kc,ti,td
681         td= 1/(alpha*(h3-b1));
682         kc= ( h2 - b0-(b1/(alpha*td)) ) / (a0+(a0/alpha));
683         ti= a0*kc/(alpha*td*h0)
684     }
685     ggbApplet.evalCommand(`C(s)=(${kc}*(1+1/(${ti}*s)+ (${td}*s /(${alpha*td} *s +1))))`);
686
687 } else if (tipoControlador === "NINGUNO"){
688     ti=0;
689     td=0;
690     kc=0;
691     h3=0;h2=0;h1=0;h0=0;
692 }

```

Figura 26: Segunda parte código en Javascript del cálculo de los parámetros kc , ti , td según el tipo de controlador

La última parte de esta etapa, correspondió a la acción de enviar los resultados a la vista algebraica de GeoGebra con el fin de mostrar la función $H(s)$ de lazo cerrado del sistema. En el caso de que el usuario no haya seleccionado ningún controlador, la $C(s)$ sería igual a 1 cómo lo muestra el paso 1 dentro del condicional de la figura 27. Esta lógica se integró dentro de un if y dentro de cada condicional se armó la función de lazo cerrado en GeoGebra utilizando el comando expand para visualizarla en forma expandida.

Como paso 2, se separó numerador y denominador de la función $H(s)$ con los comandos **Numerator** y **Denominator** propios de GeoGebra. Se extrajeron sus coeficientes y su respectivo grado utilizando los comandos **Coefficients** y **Degree**. Esto con el fin de normalizar la función en el paso 3, en el cual, se dividió los coeficientes de cada polinomio entre el coeficiente del grado mayor. Cabe resaltar que estos comandos no se pueden utilizar si los coeficientes no son números racionales.

```

716 // Calcular funcion de lazo cerrado H(s)
717 function paso1() {
718     if(tipoControlador=="NINGUNO"){
719         ggbApplet.evalCommand(`H(s)=Expand(G(s)/(1+G(s)))`);
720     }else {
721         ggbApplet.evalCommand(`H(s)=Expand(G(s)*C(s)/(1 +G(s)*C(s)))`);
722     }
723     ocultarExcepto();
724     setTimeout(paso2, 800);
725 }
726
727 function paso2() {
728     ggbApplet.evalCommand("numH = Numerator(H)");
729     ggbApplet.evalCommand("demH = Denominator(H)");
730     ocultarExcepto();
731     ggbApplet.evalCommand("coefnumH = Coefficients[numH]");
732     ggbApplet.evalCommand("coefdemH = Coefficients[demH]");
733     ggbApplet.evalCommand("gradnumH = Degree(numH)");
734     ggbApplet.evalCommand("graddemH = Degree(demH)");
735     setTimeout(paso3, 800);
736 }
737
738 function paso3() {
739     ggbApplet.evalCommand("CoefMax = Element[coefdemH, 1]");
740     ggbApplet.evalCommand("coefnumHNorm = coefnumH / CoefMax");
741     ggbApplet.evalCommand("coefdemHNorm = coefdemH / CoefMax");
742     setTimeout(paso4, 800);

```

Figura 27: Código en Javascript para la visualización de la función de lazo cerrado y su normalización

Para finalizar esta sección, en el paso 4 de la figura 28, se procedió a ordenar el polinomio de forma descendente y formar de nuevo la función de transferencia normalizada. Además, con ayuda del comando **ComplexRoot** se obtuvieron los polos del sistema y dentro de un ciclo While, se recorrió uno por uno los polos para cambiar su estilo, tamaño y visibilidad en la vista gráfica 2 de GeoGebra, utilizando comandos como **SetPointSize**, **SetColor**, **SetLineThickness** y **SetVisibleinView**, esta última permite mostrar un objeto en una vista gráfica específica. Por último, para dar tiempo a que se ejecuten todos los comandos, se utilizó un **setTimeout** y dentro de este se llamó a la función **inversa_L**, la cual se encargó de limpiar el formato de la función resultante $H(s)$, enviar una solicitud POST al backend con los datos en formato JSON, y luego esperar una respuesta del servidor, también en formato JSON. Los detalles de dicha función inversa se encuentran en los anexos.

```

745 function paso4() {
746     ordenarPolinomio("coefnumHNorm", "numH");
747     ordenarPolinomio("coefdemHNorm", "demH");
748     ggbApplet.evalCommand("Hnorm(s) = numH / demH");
749     ggbApplet.evalCommand("Delete[H]");
750     ggbApplet.evalCommand("H(s) = numH / demH");
751     ggbApplet.evalCommand("z = ComplexRoot(demH)");
752     //-----Mostrar polos en el entorno grafico 2-----
753     // Paso 1: Detectar cuántos objetos tipo z_{i} existen
754     var index = 1;
755     while (ggbApplet.exists(`z_{{index}}`)) {
756         const nombre = `z_{{index}}`;
757         // Estilo del polo "X" ( 1 es estilo en forma X)
758         ggbApplet.evalCommand(`SetPointStyle({{nombre}}, 1)`);
759         ggbApplet.evalCommand(`SetColor({{nombre}}, 255, 0, 0)`);
760         ggbApplet.evalCommand(`SetLineThickness({{nombre}}, 10)`);
761         // Mostrar en Vista Gráfica 2, ocultar en Vista Gráfica 1
762         ggbApplet.evalCommand(`SetVisibleInView({{nombre}}, 1, false)`);
763         ggbApplet.evalCommand(`SetVisibleInView({{nombre}}, 2, true)`);
764         index++;
765     }
766     ocultarExcepto();
767     ajustarGrafica2();
768     ggbApplet.evalCommand(`SetActiveView(1)`);
769     setTimeout(() => {
770         inversa_L();
771         ocultarExcepto(["Ht"]);
772     }, 400);

```

Figura 28: Código en Javascript de la función de transferencia de lazo cerrado normalizada en GeoGebra, polos del sistema y llamado de función *inversa_L*

6.1.2 Procesamiento del backend

El procesamiento simbólico del sistema de control se lleva a cabo en el backend mediante la biblioteca SymPy. Una vez que el usuario proporciona la función de transferencia $G(s)$ en la interfaz y presiona el botón de ejecución, se calcula la función de lazo cerrado $H(s)$, se normalizan sus coeficientes y se prepara para la entrada escalón mediante la expresión $H(s)/s$, la expresión de $H(s)$ es enviada al backend, donde se descompone en fracciones parciales con coeficientes simbólicos, detectando automáticamente polos reales y complejos. Se forma un sistema de ecuaciones simbólicas que se resuelve para obtener los coeficientes de la descomposición, y con ellos se aplica la transformada inversa de Laplace, tal cómo lo muestra la consola en la figura 29. El resultado, $H(t)$ junto con su valor máximo y tiempo máximo, es enviado de regreso al frontend y graficado en GeoGebra.

```

H_expr con sympify sin escalon 10/(s**2 + 2*s + 10)
H_expr (con entrada escalón): 10/(s*(s**2 + 2*s + 10))
Numerador: 10
Denominador: s*(s**2 + 2*s + 10)

Polos dict: {-1 - 3*I: 1, -1 + 3*I: 1, 0: 1}
Polos reales: [0]
Polos complejos: [-1 - 3*I, -1 + 3*I]

Fracciones parciales: [A0/s, (C0*s + D0)/((s + 1)**2 + 9)]
Suma fracc parciales:

A0/s + (C0*s + D0)/((s + 1)**2 + 9)

Numerador fracc parc expandido:

A0*s**2 + 2*A0*s + 10*A0 + C0*s**2 + D0*s

Denominador fracc parc expandido: s**3 + 2*s**2 + 10*s

Numerador fracc parc agrupado por grado:
s**2: A0 + C0
s: 2*A0 + D0
1: 10*A0

Numerador fracc parc como suma ordenada:
10*A0 + s**2*(A0 + C0) + s*(2*A0 + D0)
Coeficientes num original [10]
Coeficientes num fracc parcial: {1: 10}

Ecuaciones generadas por igualación de coeficientes:
Eq(A0 + C0, 0)
Eq(2*A0 + D0, 0)
Eq(10*A0, 10)

Fracciones parciales con valores reemplazados e inversa de Laplace:
A0 = 1.000000000000000
C0 = -1.000000000000000
D0 = -2.000000000000000

Término 1:
Fracción: 1.0/s
Inversa : 1.000000000000000

Término 2:
Fracción: (-1.0*s - 2.0)/(s**2 + 2.0*s + 10.0)
Inversa : -0.3333333333333333*exp(-1.0*t)*sin(3.0*t) - 1.0*exp(-1.0*t)*cos(3.0*t)

Respuesta temporal H(t):
1.0 - 0.3333333333333333*exp(-1.0*t)*sin(3.0*t) - 1.0*exp(-1.0*t)*cos(3.0*t)
Máximo valor: 1.3509
Ocurre en t = 1.0511 s
127.0.0.1 - - [30/Apr/2025 10:26:55] "POST /laplace HTTP/1.1" 200 -

```

Figura 29: Resultados en consola del procesamiento de la función $H(s)$ desde el backend, descomposición de fracciones parciales, formación del sistema de ecuaciones igualando los coeficientes de la ecuación característica deseada, obtención de coeficientes de las fracciones parciales y respuesta temporal $H(t)$.

A continuación, en las figuras 30, 31, 32 se detalla el código en Python del backend que procesó la expresión de la función de transferencia solicitada desde el frontend.

```

31
32 s, t = sp.symbols('s t', real=True) #Vuelve simbolica las variables s y t
33
34 def fracciones_parciales_V2(H_expr, s):
35
36     # Obtener numerador y denominador de H(s)
37     numerador = sp.numer(H_expr)
38     denominador = sp.denom(H_expr)
39
40     print("Numerador:", numerador)
41     print("Denominador:", denominador, "\n")
42
43     # Calcular los polos
44     polos_dict = sp.roots(denominador, s)
45     print("Polos dict:", polos_dict)
46
47     # Descomposición en arrays de polos
48     polos_reales = []
49     polos_complejos = []
50     for p, m in polos_dict.items():
51         if m > 1:
52             polos_complejos.extend([p] * m)
53         else:
54             if sp.im(p) == 0:
55                 polos_reales.append(p)
56             else:
57                 polos_complejos.append(p)
58
59     print("Polos reales:", polos_reales)
60     print("Polos complejos:", polos_complejos, "\n")
61
62     # Construcción de expresión simbólica para H(s) en fracciones parciales
63     coeficientes = []
64     fracciones = []
65     ecuaciones = []
66     simbolos = []
67
68     #Grado del polinomio del denominador y numerador
69     total_grado = sp.degree(denominador)
70     grado_numerador = sp.degree(numerador)
71
72     # Fracciones parciales para polos reales
73     for i, pr in enumerate(polos_reales):
74         A = sp.symbols(f"A{i}")
75         fracciones.append(A / (s - pr))
76         simbolos.append(A)
77
78     # Fracciones parciales para polos complejos
79     for i in range(0, len(polos_complejos), 2):
80         p = polos_complejos[i]
81         C = sp.symbols(f"C{i}")
82         D = sp.symbols(f"D{i}")
83         a = sp.re(p)
84         b = abs(sp.im(p))
85         fracciones.append((C * s + D) / ((s - a)**2 + b**2))
86         simbolos.extend([C, D])
87
88     print("\nFracciones parciales:", fracciones)
89     suma_fracc = sum(fracciones)
90     suma_combinada = sp.together(suma_fracc)
91     numerador_fracc = sp.expand(sp.numer(suma_combinada))
92     denominador_fracc = sp.expand(sp.denom(suma_combinada))

```

Figura 30: Primera parte código en Python de la función *inversa_L* del backend que procesa la función de transferencia de lazo cerrado.

En el backend, cómo se observa en el código de la figura 30, se comenzó por definir las variables s y t como simbólicas, se creó la función fracciones parciales, la cual, recibe cómo entrada la expresión $H(s)$ de lazo cerrado en formato string. Luego, ésta separó numerador y denominador guardados en variables diferentes mediante el comando **numer** y **denom** de Sympy. También, mediante el comando **roots** se determinaron los polos del sistema, separando cada polo en su parte real y su parte compleja.

Continuando con el procesamiento, se crearon 4 arrays cada uno para almacenar las fracciones parciales, los coeficientes simbólicos, las ecuaciones formadas al igualar con el polinomio característico deseado y los coeficientes resultantes del sistema de ecuaciones. Se ajustó el código para calcular las fracciones parciales en el caso de tener polos reales o polos complejos, mediante los comandos **sp.im** para imaginario y **sp.re** para obtener la parte real, dado que el método de descomposición de fracciones parciales se resuelve como el mencionado en el [ítem 5.4.5](#) del marco teórico para cada caso. Luego, se sumaron todos los términos simbólicos de estas fracciones.

```
112 #-----IGUALACION DE COEFICIENTES DE FUNCIONES-----
113 # Obtener los coeficientes del numerador original (ya evaluado numéricamente)
114 coef_originales = numerador.as_poly(s).all_coeffs() #Guarda los coeficientes del numerador H(s)
115 grados_originales = list(reversed(range(len(coef_originales)))) # Máximo grado hasta 0
116
117 #Crea un diccionario que asocia cada potencia s con su coeficiente numérico
118 diccionario_original = {s**g: coef_originales[i] for i, g in enumerate(grados_originales)}
119 print("Coeficientes num original", coef_originales)
120 print("Coeficientes num fracc parcial:", diccionario_original)
121
122 # Crear ecuaciones por igualación de coeficientes
123 ecuaciones = [] #Array o lista para guardar las ecuaciones
124 # Bucle for que recorre uno por uno los terminos simbolicos de numerador_agrupado (simbolico)
125 for termino, coef_simbolico in sorted(numerador_agrupado.items(), key=lambda x: -sp.degree(x[0])):
126     coef_numerico = diccionario_original.get(termino, 0) # Si no existe ese grado, igualamos a 0
127     ecuaciones.append(sp.Eq(coef_simbolico, coef_numerico)) #Construye ecuación simbólica con sp.Eq
128
129 # Mostramos las ecuaciones generadas
130 print("\nEcuaciones generadas por igualación de coeficientes:")
131 for eq in ecuaciones:
132     print(eq)
133
134 # Detectar automáticamente todas las variables simbólicas en las ecuaciones
135 incognitas = sorted(set().union(*[eq.free_symbols for eq in ecuaciones]), key=lambda x: str(x))
136
137 # Resolver el sistema de ecuaciones simbólicas
138 solucion = sp.linsolve(ecuaciones, incognitas)
139
140 # Mostrar la solución
141 print("\nFracciones parciales con valores reemplazados e inversa de Laplace:")
```

Figura 31: Segunda parte código en Python de la función *inversa_L* del backend para la igualación de coeficientes de las fracciones parciales y la ecuación característica deseada

Seguidamente, en el fragmento de código que se muestra en la figura 31, se realiza un procedimiento de **igualación de coeficientes** para resolver el sistema de manera simbólica.

Primero, se obtuvieron los coeficientes numéricos del numerador original (la función $H(s)$ en su forma estándar), organizándose desde el grado más alto hasta el grado cero. Estos coeficientes se almacenaron en un diccionario que asocia cada término s^n con su respectivo valor numérico. Luego, se compararon estos coeficientes numéricos con los coeficientes del numerador en forma simbólica (resultado de la descomposición de fracciones parciales) mediante el comando **sp.Eq** de Sympy, generando una lista de ecuaciones mediante la igualación término a término, como se muestra en la figura 31 línea 127. Cada ecuación iguala un coeficiente simbólico al valor numérico correspondiente, y si un término no está presente en el numerador original, se iguala a cero. Luego, se identifican automáticamente todas las variables simbólicas presentes en las ecuaciones (línea 135) para resolver el sistema utilizando **linsolve** de la biblioteca SymPy, obteniendo así los valores de las constantes simbólicas que hacen coincidir ambos numeradores. Este proceso es clave para determinar los coeficientes en una expansión en fracciones parciales u otras formas simbólicas equivalentes.

```

143     # Crear diccionario de soluciones con valores flotantes evaluados completamente
144     solucion_dict = {}
145     for s_ in solucion:
146         for var, val in zip(incognitas, s_):
147             val_eval = val.evalf() # No redondeamos, pero evaluamos a flotante completo
148             solucion_dict[var] = val_eval
149             print(f"{var} = {val_eval}")
150
151     inversas = [] #Array para guardar los resultados de las inversas
152     for i, fr in enumerate(fracciones): # Ciclo que recorre cada término parcial del sistema
153         try:
154             # Reemplazo y simplificación
155             fr_reemplazada = fr.subs(solucion_dict, evaluate=True) #Sustituyen constantes simbólicas por valores n
156             fr_reemplazada = sp.simplify(sp.expand(fr_reemplazada)).evalf() #Simplifica y expande la expresión
157
158             # Verificar que sólo 's' está como simbólica
159             simb_vars = fr_reemplazada.free_symbols #Verifica que sólo s sea simbolica
160             if simb_vars != {s}: #Condicional en caso de error
161                 raise ValueError(f"Término {i}: variables simbólicas no válidas: {simb_vars}")
162
163             print(f"\nTérmino {i+1}:") #Muestra en consola el n termino
164             print(f"Fracción: {fr_reemplazada}") #Muestra en consola los resultados
165
166             inversa = sp.inverse_laplace_transform(fr_reemplazada, s, t) #Aplica la transformada de laplace
167             inversa = inversa.subs(sp.Heaviside(t), 1) #Sustituye la funcion Heaviside por 1 para t > 0
168             inversas.append(inversa) #Se agrega la respuesta del n termino de inversa a list inversas
169             print(f"Inversa : {inversa}") #Muestra en consola la inversa del n termino
170
171         except Exception as e:
172             print(f"Inversa : Error en término {i}: {e}") #Muestra en consola error en caso de no calcularlo
173

```

Figura 32: Tercera parte código en Python de la función *inversa_L* de la solución al sistema de ecuaciones de los coeficientes de las fracciones parciales y respuesta temporal del sistema

Finalmente, en esta etapa del procedimiento, se resolvió el sistema de ecuaciones generado por la igualación de coeficientes, obteniendo los valores numéricos de las constantes simbólicas presentes en la expansión en fracciones parciales. Seguidamente, se sustituyeron dichos valores numéricos en cada término parcial de la función de transferencia, asegurando que la única variable simbólica restante fuese “ s ”, como se muestra en las líneas 158 a 161 de la figura 32. Posteriormente, se aplicó la transformada inversa de Laplace a cada uno de estos términos, utilizando SymPy, y se reemplazó la

función escalón de Heaviside por la constante 1 para simplificar la expresión final. En caso de que alguna fracción no fuese válida para la transformación, se implementó un control de errores para identificar y reportar el término problemático, garantizando la robustez del proceso.

Cabe mencionar, que en caso de que la respuesta simbólica no pueda obtenerse —por ejemplo, si la expresión es demasiado compleja— el backend puede generar una solución numérica basada en puntos discretos. Esta solución también es enviada a la interfaz y graficada como una curva aproximada. Finalmente, el sistema detecta automáticamente el valor máximo de la respuesta y el instante en que ocurre, lo que permite verificar si se cumplen los criterios de diseño establecidos por el usuario.

6.2. Método del lugar geométrico de las raíces:

Para implementar el método del lugar geométrico de las raíces en GeoGebra, se comenzó por seguir la lógica que se muestra en el diagrama de flujo de la figura 33. En la cual, el usuario comienza por ingresar los coeficientes del numerador y denominador de la función de transferencia $G(s)$ de planta, mediante cajas de entrada en la interfaz web. Estos datos se almacenan en variables y se crea un slider en GeoGebra llamado K_c , el cual cuenta con un rango de ganancias proporcionales para que el usuario decida en qué valor dejar dicha ganancia, realizando un primer supuesto de tener un controlador netamente proporcional. El usuario da un primer clic en el botón iniciarLGR que se muestra en la figura 38, y el evento llama a la función **“IniciarLGR”** de Javascript (ver figura 34). Esta función se encarga de crear en GeoGebra las funciones de transferencia $G(s)$ de planta, $C(s)$ del controlador y $H(s)$ de lazo cerrado. Con base en esta última, se calculan los polos del sistema y se grafica en GeoGebra el lugar geométrico de las raíces. Justo después, esta misma función de Javascript hace la solicitud al backend, llamando a la función **“inversa_L”** (ver figura 30), para que le devuelva la respuesta temporal del sistema. Con un primer gráfico del LGR y la respuesta temporal, el usuario decide si el controlador proporcional es suficiente para ubicar los polos dominantes donde él lo requiere (símbolo de decisión del diagrama de flujo). Si la decisión es negativa, el usuario puede seleccionar a través de un desplegable otro tipo de controlador, ya sea PI, PD, o PID. Con lo cual, se desbloquean las casillas de entrada T_i , T_d o α dependiendo el caso. Seguidamente, el usuario ingresa sus valores y actualiza el LGR al clic del botón **“ActualizarControlador”**. Con lo cual el flujo vuelve al punto de llamado de la función **“IniciarLGR”**; se grafican de nuevo el LGR y la respuesta temporal, y el flujo retorna al símbolo de decisión, el cual vuelve y verifica si el sistema cumple con los requerimientos deseados. El código en Javascript y la explicación detallada de este método se muestra en el ítem 6.2.1

A continuación, se muestra en la figura 33 el diagrama de flujo que contiene la lógica implementada en este método.

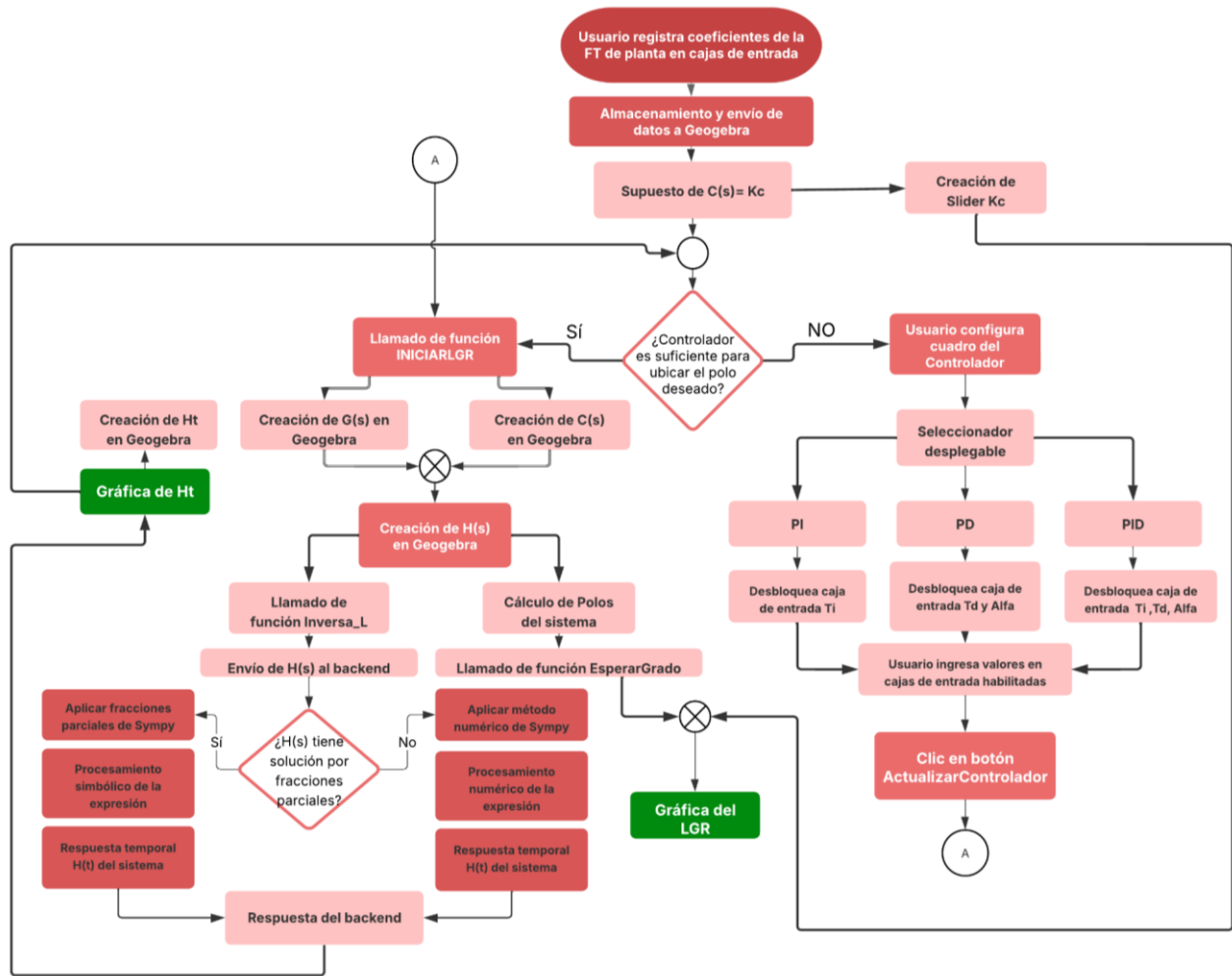


Figura 33 : Diagrama de flujo de caso de uso para el método del lugar geométrico de las raíces

Con la lógica del diagrama de la figura 33, a continuación se explican el procesamiento del frontend y backend de este método. Se le llama frontend a todo el código en Javascript que permitió mostrar la interfaz web, su diseño y el cual realizó comunicación directa con la API de GeoGebra. Backend se le llama al código realizado en Python que se encargó de procesar la expresión $H(s)$ de lazo cerrado y devolver su respuesta temporal.

Con el objetivo de comparar resultados más adelante y verificar la confiabilidad de esta aplicación web, se utilizó por defecto y como prueba, la función de transferencia de planta que se muestra a continuación:

$$G(s) = \frac{0,077 s + 154,535}{s^2 + 12,0318 s + 154,287} \quad (2)$$

La cuál, es la misma función de planta del artículo que se muestra en la referencia [24]. Este implementa el mismo método en el entorno de SisoTool de Matlab. También fueron utilizados los mismos parámetros del controlador PID.

6.2.1 Procesamiento del frontend

Como se mencionó anteriormente, el usuario comienza por registrar los coeficientes de $G(s)$ en cajas de entradas, pero sólo se guardan en variables hasta el momento en que se presiona el botón Iniciar LGR, el cual llama a la función “*iniciarLGR()*” de la figura 34. Esta función primeramente verifica si ya existe el slider K_c en GeoGebra, de lo contrario tiene una espera de 2 segundos y vuelve a llamar la función. Esto se realiza, debido a que es un parámetro indispensable en el cálculo del LGR y si no existe en GeoGebra devuelve error. Seguidamente, se guardan los coeficientes de $G(s)$ en variables a_2, a_1, a_0 para el numerador en orden descendente y b_2, b_1, b_0 para el denominador, con ello se arma la función y se envía a GeoGebra utilizando *evalCommand*, cómo se muestra en la línea 691 de la figura 34.

```

668 function iniciarLGR(){
669     // -----Esperar a que GeoGebra registre los sliders-----
670     if (!ggbApplet.exists("Kc")) {
671         console.warn("⚠ Esperando sliders...");
672         setTimeout(iniciarLGR, 200); // vuelve a intentar más tarde
673         return;
674     }
675     //-----Almacenamiento en variables de los coeficientes de G(s)-----
676     ordenNum = numeradorG.length - 1; //Variable orden de asignacion numerador
677     ordenDen = denominadorG.length - 1; //Variable orden de asignacion denominador
678     a2 = (ordenNum >= 2 ? numeradorG[ordenNum - 2] : 0); //Coeficiente de s^2
679     a1 = (ordenNum >= 1 ? numeradorG[ordenNum - 1] : 0); //Coeficiente de s^1
680     a0 = (numeradorG[ordenNum]); //Coeficiente de s^0
681
682     b2 = (ordenDen >= 2 ? denominadorG[ordenDen - 2] : 0); //Coeficiente de s^2
683     b1 = (ordenDen >= 1 ? denominadorG[ordenDen - 1] : 0); //Coeficiente de s^1
684     b0 = (denominadorG[ordenDen]); //Coeficiente de s^0
685     console.log("coef. numerador G= ",a2, a1, a0,"coef. denominador G= ",b2, b1, b0);
686
687     // -----Definir funciones G(s)-----
688     ggbApplet.evalCommand(`G(s) = (${a2}*s^2 + ${a1}*s + ${a0})/(${b2}*s^2 + ${b1}*s + ${b0})`);
689     console.log(`→ G(s): (${a2}*s^2 + ${a1}*s + ${a0}) / (${b2}*s^2 + ${b1}*s + ${b0})`);
690
691     actualizarC(); //Llamado de funcion actualizar C para calcular controlador
692     ggbApplet.evalCommand(`H(s) = Expand((G*C) / (1 + (G * C)))`); //H(s) de lazo cerrado
693
694     // Definir numerador y denominador de H en GeoGebra
695     ggbApplet.evalCommand(`B(s) = (Numerator(H(s)))`); //Numerador de H(s)
696     ggbApplet.evalCommand(`A(s) = (Denominator(H(s)))`); //Denominador de H(s)

```

Figura 34: Función iniciarLGR en Javascript del frontend para el método LGR

Luego, cómo lo muestra el diagrama de flujo, lo siguiente que hace la función *iniciarLGR* después de crear $G(s)$ es crear $C(s)$ del controlador en GeoGebra. Para ello se llama a la función “*actualizarC*” de la figura 35, la cual se encarga de construir dinámicamente la expresión simbólica del controlador $C(s)$ en función del tipo de controlador seleccionado (NINGUNO, PI, PD o PID) y los parámetros ingresados por el usuario (T_i , T_d y α). Primero verifica si hay un tipo de controlador definido; luego obtiene los valores numéricos desde los campos de entrada del HTML. Dependiendo del tipo de controlador, genera la expresión

simbólica correspondiente para $C(s)$, considerando en el caso del PID un filtro derivativo genérico. Esta expresión se envía a GeoGebra mediante `ggbApplet.evalCommand`, se actualiza en el display mediante la función “*actualizarCT_display*” (detallada en los anexos), y si ocurre algún error, se muestra un mensaje de alerta. Además, se realiza un pequeño ajuste numérico (cambio diminuto) a la constante K_c para forzar la actualización del lugar geométrico de las raíces (LGR), y gestiona el registro de un listener para que la función se actualice automáticamente si cambia el valor de K_c .

```

606 function actualizarC() {
607     if (!tipoControlador) return;
608     //-----Obtener valores directamente de los input HTML-----
609     Ti = parseFloat(document.getElementById("TiInput").value) || 1;
610     Td = parseFloat(document.getElementById("TdInput").value) || 0;
611     alpha = parseFloat(document.getElementById("TfInput").value) || 0;
612
613     //-----Crear expresion C(s) segun el tipo de controlador-----
614     let C_expr;
615     if (tipoControlador === "NINGUNO") { // Caso de Controlador constante
616         ggbApplet.setValue("Kc", 1);
617         C_expr = "Kc"
618     } else {
619         // Construir expresión del controlador
620         C_expr = `Kc * (1`;
621         if (tipoControlador === "PI") { //Caso de controlador PI
622             C_expr += ` + 1/(${Ti}*s)`;
623         } else if (tipoControlador === "PD") { //Caso del controlador PD
624             C_expr += ` + (${Td}*s)/(${alpha}*${Td}*s + 1)`;
625         } else if (tipoControlador === "PID") { //Caso del controlador PID
626             C_expr += ` + 1/(${Ti}*s) + (${Td}*s)/(${alpha}*${Td}*s + 1)`;
627         }
628         C_expr += `)`;
629     }
630     try {
631         ggbApplet.evalCommand(`C(s) = ${C_expr}`); //Envío a geogebra de C(s)
632         actualizarCT_Display(); //Actualizacion en display
633         ocultarExcepciones(); //Ocultar objetos innecesarios
634     } catch (e) {
635         alert("⚠ Error al evaluar C(s): " + e.message);
636     }
637     console.log("C(s) =", C_expr);
638
639     //-----TRUCO para forzar actualización del LGR-----
640     //Detectar evento o cambio en Kc con Listener
641     if (listenerRegistrado) {
642         ggbApplet.unregisterObjectUpdateListener("Kc", actualizarC);
643         listenerRegistrado = false;
644     }
645
646     //-----Pequeño cambio en Kc para forzar actualizacion de LGR
647     const kc_actual = ggbApplet.getValue("Kc");
648     ggbApplet.setValue("Kc", kc_actual + 1e-10);
649     ggbApplet.setValue("Kc", kc_actual);
650
651     if (!listenerRegistrado) {
652         ggbApplet.registerObjectUpdateListener("Kc", actualizarC);
653         listenerRegistrado = true;
654     }
655 }

```

Figura 35: Función actualizarC para formar la expresión del controlador en GeoGebra

Posteriormente, ya con las función $G(s)$ y $C(s)$ se crea la función de lazo cerrado $H(s)$ cómo lo indica el diagrama de flujo de la figura 33 y se envía a GeoGebra . Además, se extrajeron numerador y denominador cómo lo muestra la línea 695 (ver nuevamente figura 34) . Por último, se realiza una espera para verificar que el polinomio característico $A(s)$ de lazo cerrado exista antes de calcular los polos del sistema. Esta verificación se realiza mediante la función “*esperarGrado*” de la figura 36, con un máximo de 20 intentos. Luego de verificar la existencia de $A(s)$, se procede a extraer su grado mediante el comando *Degree* y se almacena; también se realiza una limpieza de los datos para evitar residuos anteriores, eliminando gráficos y polos que causen errores por repetición de estos. Seguidamente, en la línea 715 se extrajeron los polos del polinomio característico con el comando ***ComplexRoot(A)*** de GeoGebra. Por último, se creó un ciclo *for* para dibujar las ramas del LGR, mediante el uso del comando ***Locus*** (línea 730), el cual recibe cómo parámetros los polos calculados anteriormente y el slider K_c , el cual se creó desde el inicio del cargue de la interfaz web. Adicional a esto, se le dio formato y color a las ramas y polos del sistema.

A continuación, se muestra en la figura 36 el detalle de la función “*esperarGrado*” encargada de calcular y dar formato a las ramas del LGR.

```

690 // Función para esperar a que grado esté disponible
691 function esperarGrado(intentos = 0) {
692     if (intentos > 20) { //Condicional de limite de intentos si no existe A
693         console.error("Demasiados intentos esperando A(s). Abortando...");
694         return;
695     }
696     const existeA = ggbApplet.exists("A"); //Variable booleana de la existencia de A
697     if (!existeA) { //Condicional para verificar la existencia de A
698         console.warn("A(s) aún no existe. Reintentando...");
699         setTimeout(() => esperarGrado(intentos + 1), 300);
700         return;
701     }
702     const grado = ggbApplet.getValue("Degree(A)"); //Variable que obtiene el grado de A
703     if (!isNaN(grado) && grado > 0) { //Condicional para verificar el grado
704         console.log(" Grado de A(s):", grado);
705     }
706     // -----Limpiar residuos anteriores-----
707     const gradoMax = 10;
708     for (let i = 1; i <= gradoMax; i++) {
709         if (ggbApplet.exists(`LGR_${i}`)) ggbApplet.evalCommand(`Delete[LGR_${i}]`);
710         if (ggbApplet.exists(`z_${i}`)) ggbApplet.evalCommand(`Delete[z_${i}]`);
711         if (ggbApplet.exists(`p_${i}`)) ggbApplet.evalCommand(`Delete[p_${i}]`);
712     }
713     //-----Calculo de los polos del sistema de lazo cerrado-----
714     ggbApplet.evalCommand(`ComplexRoot(A)`);
715 }

```

Figura 36: a) Primera parte función *esperarGrado* que verifica la existencia del polinomio característico de lazo cerrado y calcula los polos del sistema

```

727     for (let i = 1; i <= grado; i++) { //Bucle para dibujar cada rama del LGR
728
729         //Calculo del LGR CON LOCUS
730         ggbApplet.evalCommand(`LGR_${i} = Locus(z_${i}, Kc)`); //Comando LOCUS
731
732         //Estilo de las ramas y los polos
733         const [r, g, b] = colores[(i - 1) % colores.length]; //Color existente
734         ggbApplet.setColor(`LGR_${i}`, r, g, b); //Color de la rama
735         ggbApplet.setLineThickness(`LGR_${i}`, 5); //Grosor de la rama
736         ggbApplet.evalCommand(`SetPointSize(z_${i}, 1)`); //Estilo X del polo
737         ggbApplet.evalCommand(`SetLineThickness(z_${i}, 15)`); //Grosor del polo
738
739     }
740     ocultarExcepto(); //Funcion para ocultar objetos innecesarios
741     inversa_L(); //Llamado de funcion inversa_L para calcular H en el tiempo
742
743 } else {
744     console.warn("⚠ Esperando que A(s) esté lista... intento", intentos);
745     setTimeout(() => esperarGrado(intentos + 1), 300);
746 }
747 }
748 // Llamar a la función ESPERARGRADO para calcular LGR con A existente
749 esperarGrado();
750
751 }

```

Figura 36: b) Segunda parte función esperarGrado que dibuja el lugar geométrico de las raíces y llama la función inversa_L

7. RESULTADOS OBTENIDOS

7.1. Resultados del método de asignación de polos

Los resultados del método de asignación de polos propuesto en el entorno web embebido en GeoGebra, utilizando la misma planta y parámetros de diseño del artículo de referencia [23], el cual realiza control a la velocidad de un motor DC; arrojaron visualmente las gráficas de la respuesta temporal $H(t)$ de la función de lazo cerrado del sistema, sus polos y los parámetros del controlador K_c , T_i , T_d que se muestran en la figura 37. También se obtuvo la función de transferencia de lazo cerrado $H(s)$, la función de planta $G(s)$ y del controlador $C(s)$, estas dos últimas se observan en el display de la interfaz.

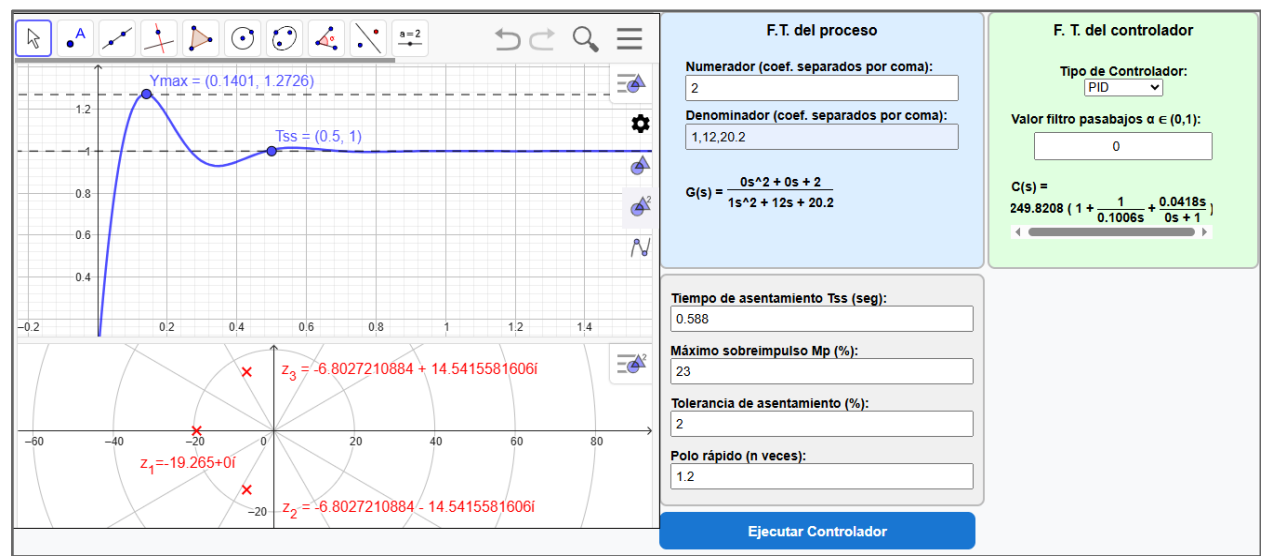
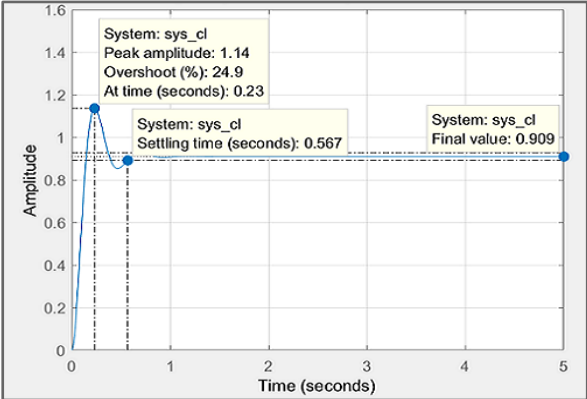
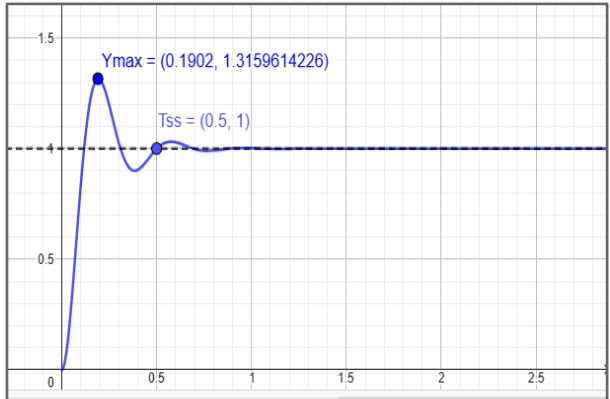


Figura 37: Caso a) Resultados del método propuesto de asignación de polos con su respuesta temporal, polos del sistema y condiciones de diseño en $T_{ss} = 0,588$ seg $M_p = 23\%$, Tolerancia 2% del mismo proceso aplicado en el artículo [23]

Tabla 1: Comparación de respuestas temporales ante una entrada escalón para distintos tipos de controladores, tiempos de asentamiento y sobreimpulso entre método del artículo [23] y método propuesto de asignación de polos

Método del artículo [23] graficado en Matlab	Método propuesto de asignación de polos graficado en GeoGebra
Caso a): Controlador P con $K_p=100$, $M_p=24,9\%$, $T_{ss}=0,567$ seg	Caso a): Controlador P con $K_p=142$, $M_p=31\%$, $T_{ss}=0,5$ seg
	
Caso b): Controlador PID con $K_p=100$, $K_i=200$, $K_d=10$, $M_p=1,03\%$ y $T_{ss}=0,257$ seg	Caso b): Controlador PID con $K_p=189$, $K_i=236$, $K_d=9,5$ $M_p=1,03\%$ y $T_{ss}=0,22$ seg

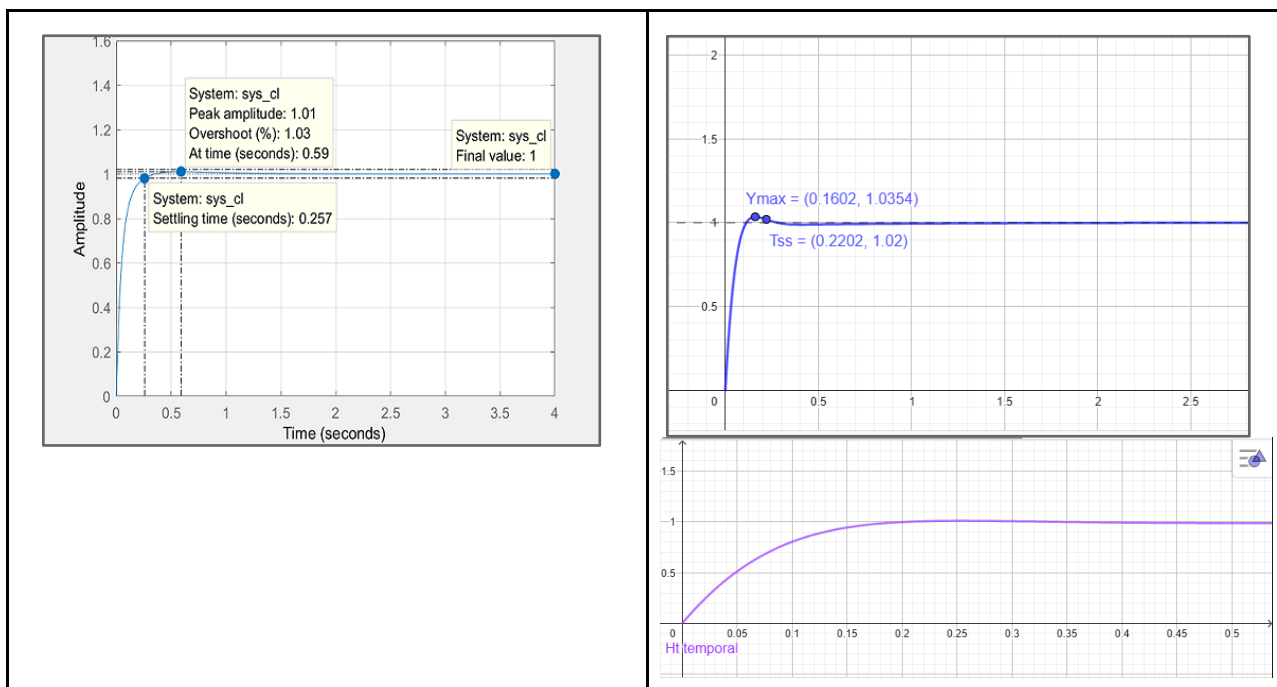


Tabla 2:
*Resumen de
parámetros entre el
método del artículo
[23] y el método
propuesto de
asignación de polos*

CASO	PARÁMETRO	MATLAB	GEOGEBRA PROPUESTO	DIFERENCIA (%)
Caso A	Kp	100	142	42 %
	Mp	25	31	24 %
	Ts	0.56	0.5	11 %
Caso B	Kp	100	189	89 %
	Ki	200	236	18 %
	Kd	10	9.5	5 %
	Mp (%)	1.03	1.029	0,1 %
	Ts (s)	0.25	0.22	12%

7.2. Resultados del método del lugar geométrico de las raíces:

El resultado de correr el código del ítem 6.2 se muestra en la figura 38, en la cual se observa con 3 colores distintos los 3 caminos que toman los polos del sistema, los cuales se representan con una X en la gráfica de la interfaz. En el lado izquierdo de la figura, se muestran los datos de la función de planta $G(s)$ y el tipo de controlador $C(s)$ seleccionado, cómo sus parámetros de ganancia proporcional K_c , tiempo integral T_i y tiempo derivativo T_d para este caso. Adicionalmente, se muestra una comparación con los datos obtenidos del artículo “*Diseño de un Controlador PID Difuso para una planta de Segundo Orden Sub amortiguada usando LABVIEW*” de la referencia [24], para dos casos de controladores y la misma planta. La comparación para el caso 1 se muestra en la tabla 2.

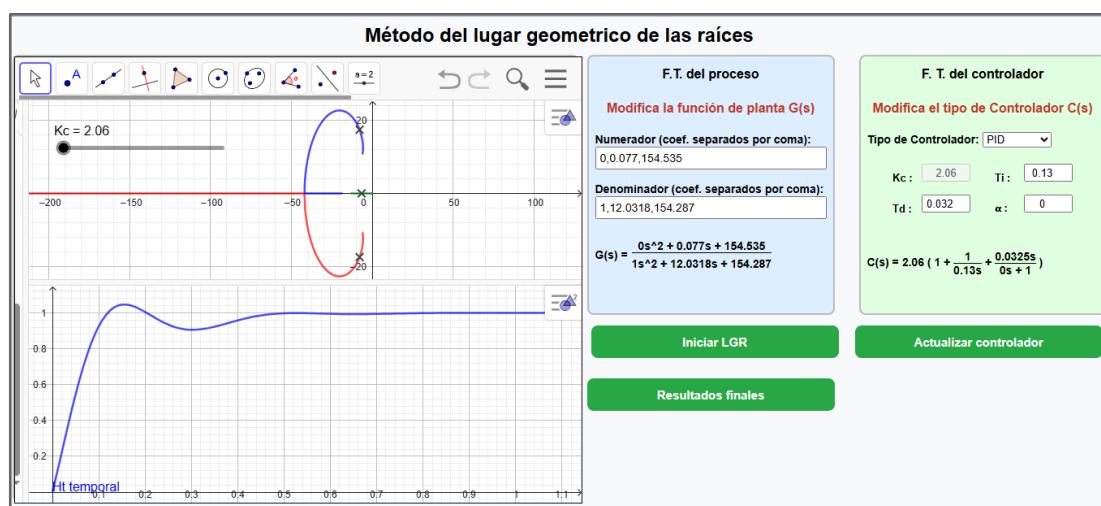


Figura 38: Caso 1) Resultados del método propuesto del LGR en la interfaz web, con un controlador PID robusto y parámetros $K_p=2,058$ $K_i=15,83$ $K_d=0,0669$, mismo sistema utilizado en la figura 20 del artículo [18]

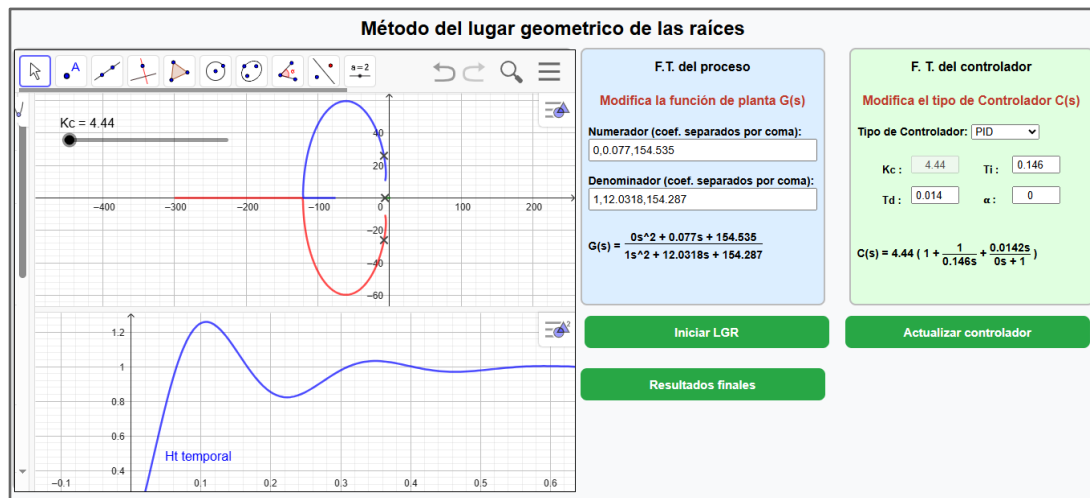
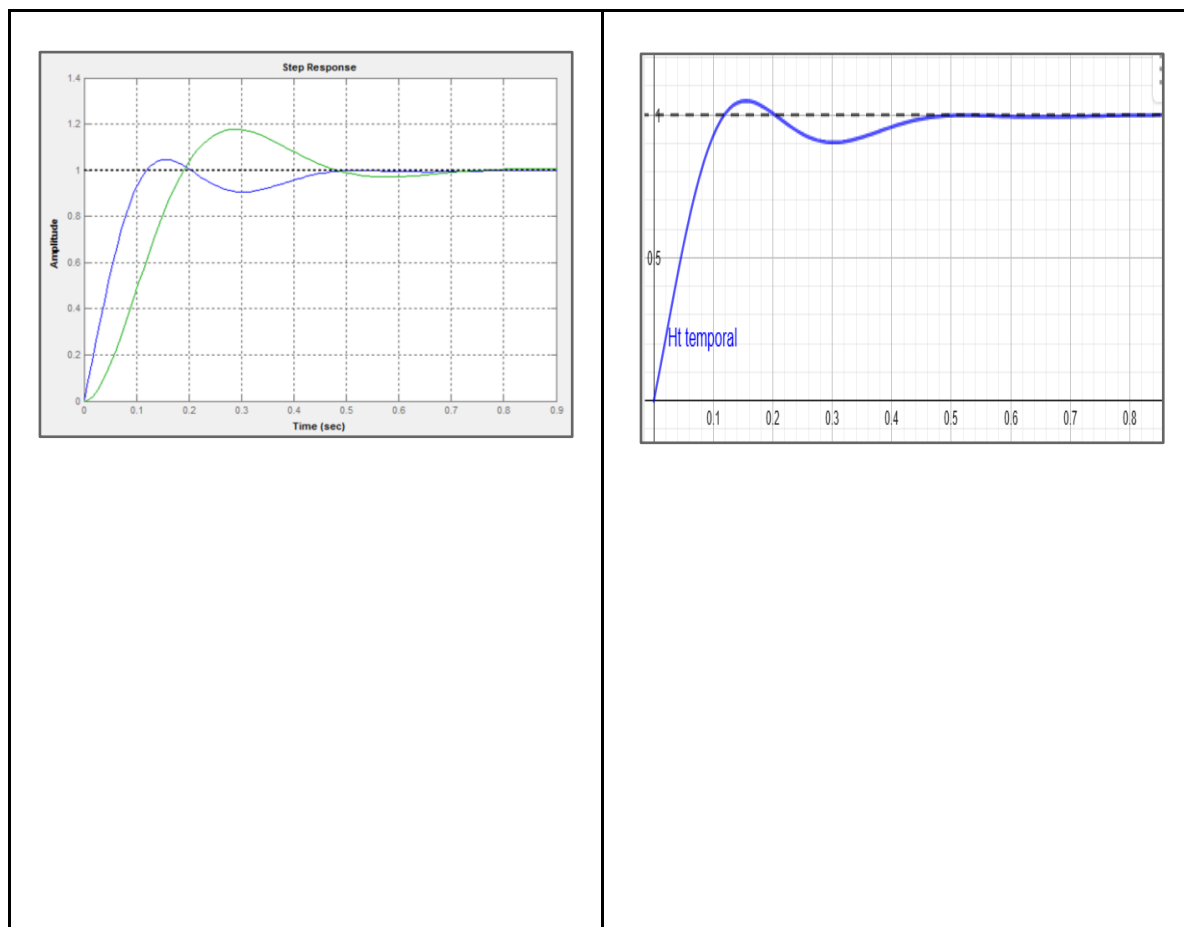


Figura 39: Caso 2) Resultados del método propuesto del LGR en la interfaz web con controlador PID (Z-N) y parámetros $K_p=4,44$ $K_i=30,4$ $K_d=0,063$, mismo sistema utilizado en la figura 20 del artículo [18]

Tabla 3: Comparación del lugar geométrico de las raíces y respuesta temporal del caso 1 para un controlador PID robusto entre el método del artículo [18] y el método propuesto del LGR

CONTROLADOR PID ROBUSTO (Caso 1 con $K_p=2,058$ $K_i=15,83$ $K_d=0,0669$)	
Método del artículo [18] graficados en SisoTool	Método propuesto del LGR graficado en GeoGebra
<i>Lugar Geométrico de las raíces</i>	<i>Lugar Geométrico de las raíces</i>
Respuesta temporal en azul de la función de lazo cerrado ante una entrada escalón	Respuesta temporal de la función $H(s)$ de lazo cerrado ante una entrada escalón

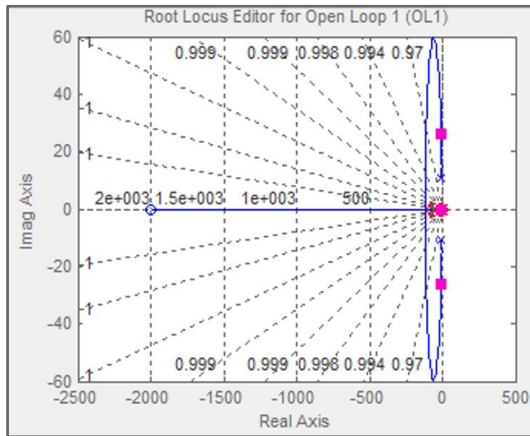


En el artículo [18], también se utilizó otro controlador con la misma planta, llamado en el artículo como controlador PID Z-N, ya que se aplicó el método de Ziegler y Nichols. Los resultados obtenidos de implementar en la interfaz web las mismas condiciones de este controlador y su planta se muestran en la figura 39; además, de su respectiva tabla comparativa entre el método del artículo y el método propuesto para los gráficos del LGR y sus respuestas temporales (tabla 3).

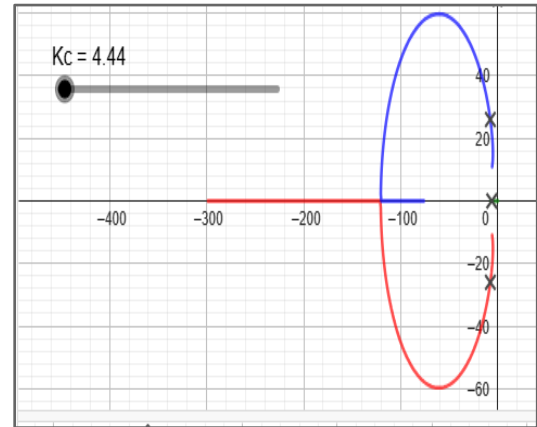
Tabla 4: Comparación del lugar geométrico de las raíces y respuesta temporal del caso 2 para el controlador PID Z-N entre el método del artículo [18] y el método propuesto del LGR

CONTROLADOR PID Z-N (Caso 2 con $K_p=4,44$ $K_i=30,4$ $K_d=0,063$)	
Método del artículo [18] graficados en SisoTool	Método propuesto del LGR graficado en GeoGebra

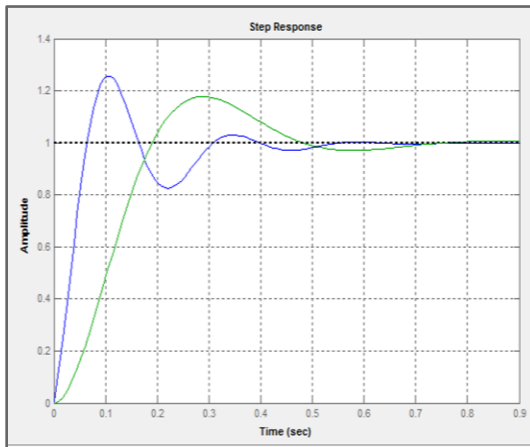
Lugar Geométrico de las raíces



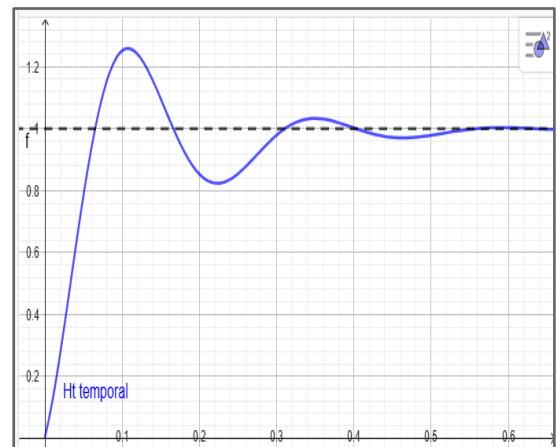
Lugar Geométrico de las raíces



Respuesta temporal en azul ante una entrada escalón de la función de lazo cerrado



Respuesta temporal ante una entrada escalón de la función de lazo cerrado



8. DISCUSIÓN DE RESULTADOS

La comparación de los resultados logrados a través de la herramienta creada en GeoGebra en comparación a los mencionados en las publicaciones de referencia [17] y [18] revela la validez y eficacia de las estrategias propuestas. En relación al enfoque de asignación de polos se consiguió reproducir tanto los comportamientos temporales esperados como el tiempo de establecimiento T_{ss} y el sobreimpulso M_p de un sistema de control de velocidad para un motor DC utilizando los mismos parámetros que se emplearon en [17], aunque adaptados al entorno gráfico proporcionado por GeoGebra. Se ha observado que este método de asignación de polos es muy sensible ante cambios en la posición del polo rápido ω_r que el usuario introduce en la casilla “múltiplo del polo rápido” en la interfaz. Cuando un polo se coloca cerca del eje imaginario se nota un incremento de tiempo de establecimiento y una disminución en la velocidad de respuesta que podría afectar el rendimiento dinámico del sistema. Por otra parte, al situar un polo lejos del eje imaginario en exceso en la parte real negativa, se notó una aparentemente mejoría en la velocidad, pero esto derivó en una ganancia mayor para el controlador a expensas de posibles efectos amplificados del ruido y saturaciones. Asimismo, si se coloca un polo en una posición incorrectamente o asimétrica, la respuesta puede volverse menos estable. La presencia de una interferencia no deseada sugiere que el enfoque de asignación de polos es claro y metódico; sin embargo, este depende en gran medida de la selección correcta de las ubicaciones de polos deseadas. Esta particularidad subraya la importancia de llevar a cabo un análisis minucioso de los requisitos de rendimiento y asegurar la resistencia del sistema frente a pequeñas alteraciones en los parámetros.

Además de eso cuando se agregó un polo mediante el filtro derivativo α un incremento en este ajuste causó que la acción derivativa se volviera más agresiva resultando en una respuesta más rápida pero también más susceptible al ruido, lo cual ocasionó oscilaciones indeseadas en ciertos escenarios. Esta mayor sensibilidad disminuyó la robustez del sistema demostrando que un valor elevado de α compromete la estabilidad y la suavidad de la señal de control. Por el contrario, cuando se utilizan valores bajos de α , la respuesta es más consistente y menos influenciada por perturbaciones de alta frecuencia, aunque la acción derivativa se ve disminuida. Cabe resaltar que seleccionar un correcto valor del filtro derivativo implica un equilibrio crucial entre la velocidad de respuesta y la capacidad de rechazar ruido.

Por lo tanto en este procedimiento fue vital tanto la posición del polo rápido como la del polo adicional originado a través del filtro derivativo. Dado que se trata de una herramienta de interactividad estos aspectos están ampliamente influenciados por el criterio propio del usuario al introducir los valores respectivos. Al final se notó que a pesar de que los valores del controlador diferían levemente en comparación al artículo de referencia, las respuestas temporales obtenidas fueron similares o incluso mejoradas en ciertos casos. Especialmente notable fue el caso c), donde se redujo el sobreimpulso M_p al 1.03 % y el tiempo de establecimiento T_{ss} a 0.22 segundos, en contraste a los 0.25 segundos mencionados en el artículo de referencia.

Por otro lado, en relación al enfoque del método de lugar geométrico de las raíces (LGR), la herramienta desarrollada también facilitó una representación visual clara de un sistema lineal que utiliza un controlador PID basado en el artículo [18]. Este sistema se basó en un circuito TL082 (un amplificador operacional JFET dual), utilizado como planta física. Esta representación permitió observar cómo los polos del sistema se desplazan y cómo responde ante cambios en la ganancia K_c . En comparación con los resultados obtenidos mediante el uso de SisoTool (mencionado en [18]), la representación visual y precisión del método aplicado en GeoGebra son satisfactorios y además

ayudan en la comprensión del impacto de los controladores PID robustos y PID-ZN que se utilizan como verificación de este método. Las tablas comparativas mostraron consistencia en los resultados particularmente en la ubicación de los polos y la respuesta al escalón lo cual respalda la eficacia de GeoGebra como una herramienta complementaria en automatización y control.

Cabe destacar que, en ambos métodos, se optó por utilizar un respaldo de un backend desarrollado en Python para procesar la expresión de lazo cerrado del sistema y llevar a cabo cálculos simbólicos con el fin de enviar la respuesta temporal al frontend de la interfaz. Esta decisión se tomó porque resultó complicado establecer una comunicación eficaz desde el código hacia el entorno simbólico de GeoGebra. A pesar de que hay funciones como *evalCommandCAS()*, que están pensadas para comunicarse con el sistema CAS (Sistema de Álgebra Computacional) desde JavaScript; estas funciones no operaron correctamente durante la ejecución del código. Específicamente al utilizar *evalCommandCAS()*, no se obtenía ninguna respuesta efectiva del software simbólico; lo cual limitó la funcionalidad de GeoGebra al ámbito algebraico y gráfico.

Este inconveniente fue comunicado al equipo de desarrolladores de GeoGebra sin recibir una solución o respuesta que permitiera hacer uso correcto del entorno CAS desde JavaScript. Dada esta limitación y para no detener el avance del proyecto se decidió incorporar el procesamiento simbólico en Python para procesar la función de lazo cerrado, aplicar el método de fracciones parciales y obtener la respuesta temporal del sistema, aprovechando herramientas como SymPy que brindan un sólido respaldo para este tipo de operaciones.

Sin embargo, esto no implica que GeoGebra deje de ser una herramienta poderosa para el desarrollo de aplicaciones en el ámbito del control. Por el contrario, se ha evidenciado su potencial como entorno interactivo, y se tiene la esperanza de que en futuras actualizaciones, los problemas de comunicación con el entorno CAS puedan ser solucionados. Además, se ha comprobado que GeoGebra se puede integrar perfectamente con otros lenguajes de programación como Python, lo cual amplía sus capacidades y opciones de aplicación.

Ambas metodologías propuestas en este proyecto presentaron beneficios desde el punto de vista educativo y de validación de conceptos al permitir que el usuario pueda visualizar de manera instantánea cómo cambian los parámetros de un sistema y su respuesta dinámica. Este enfoque visual e interactivo sirve como un vínculo entre la teoría y la práctica, especialmente en contextos donde se busca aprovechar herramientas de desarrollo gratuitas y accesibles para el diseño de controladores en sistemas reales.

9. CONCLUSIONES

- Se implementaron dos métodos de forma exitosa para el desarrollo y evaluación de controladores PID, tales como la asignación de polos y el LGR (locus de raíces), dentro del contexto algebraico y gráfico proporcionado por GeoGebra. Estos métodos permitieron la reproducción y validación de resultados mencionados en publicaciones científicas reconocidas.
- El enfoque de asignación de polos facilitó la creación de controladores cuyas respuestas temporales cumplen los criterios de desempeño establecidos por el usuario, cómo el sobrepaso máximo, con una diferencia del **24%** y una diferencia en tiempo de establecimiento de **12%**, entre el método propuesto y los resultados del artículo [17] para los casos que más se alejaron de los resultados esperados. En dichos casos, se encontró una sensibilidad muy alta respecto al parámetro encontrado de **Kp** cuyos porcentajes de diferencia fueron muy elevados y se sugiere hacer un ajuste para precisar la sensibilidad del método, el cual se ve afectado por el polo rápido ingresado. El ajuste puede ser un límite para ingresar este polo, con el fin de no afectar la estabilidad del método. Sin embargo, el comportamiento gráfico en todos los casos resultó bastante similar al logrado en MATLAB, validando así la efectividad de la implementación.
- Se demostró que la efectividad del método sugerido de asignación de polos está condicionado por la localización de los mismos; en particular el “polo rápido”, cuya mala ubicación podría comprometer la estabilidad, velocidad o robustez de la respuesta temporal; esta susceptibilidad también señaló que el método, aunque sistemático, demanda criterio por parte del diseñador y análisis adicional de robustez.
- El enfoque utilizado por el método del LGR no solo permitió una representación gráfica en tiempo real de los desplazamientos de los polos sino que también simplificó la comparación contra herramientas especializadas como SisoTool y obtuvo resultados consistentes asegurando la fiabilidad del sistema.
- GeoGebra, a pesar de no ser una herramienta típicamente empleada en la automatización industrial, se mostró como una opción sólida y efectiva al fusionar JavaScript y funciones gráficas, abriendo nuevos horizontes en entornos educativos y de prototipos rápidos.
- Se concluye que la estrategia de desarrollar un backend en Python junto con la librería SymPy como solución híbrida fue efectiva para superar la incapacidad de utilizar directamente el entorno simbólico (CAS) de GeoGebra desde JavaScript. Esta incorporación resolvió la limitación técnica y demostró que GeoGebra puede ser complementado por otros lenguajes y herramientas para ampliar su utilidad en el diseño y análisis de sistemas de control.
- Se concluye que esta herramienta puede ser útil tanto para propósitos educativos como en la fase inicial de desarrollo de controladores, sobre todo en entornos donde el acceso a software costoso es fundamentalmente limitado.

10. IMPACTO DEL PROYECTO

El desarrollo de esta herramienta presenta un impacto significativo en diversos ámbitos. En el ámbito académico, permite a estudiantes y docentes de ingeniería visualizar en tiempo real el comportamiento dinámico de sistemas de control, facilitando la comprensión de conceptos como la asignación de polos, el lugar geométrico de las raíces y la respuesta temporal de sistemas de segundo orden. Al estar construida en el motor matemático de GeoGebra, una plataforma gratuita y ampliamente difundida, la herramienta democratiza el acceso a simulaciones avanzadas sin necesidad de licencias costosas como las de MATLAB o LabVIEW.

Desde el punto de vista tecnológico, el proyecto demuestra la viabilidad de integrar lenguajes de programación como JavaScript y Python con entornos gráficos interactivos, ampliando las capacidades tradicionales de GeoGebra. Esto abre la puerta a nuevas aplicaciones en automatización, control y modelado de sistemas dinámicos dentro de plataformas educativas no convencionales.

En cuanto al impacto económico, la solución propuesta se basa en software libre, lo que la convierte en una alternativa de bajo costo para instituciones educativas con recursos limitados, permitiéndoles acceder a herramientas de diseño y análisis que normalmente requieren infraestructura de software propietaria.

11. RESULTADOS DE APRENDIZAJE ALCANZADOS

A continuación, se enumeran los resultados de aprendizaje alcanzados durante el desarrollo del presente proyecto de grado:

1. Aplicó conceptos fundamentales de teoría de control, como la asignación de polos y el lugar geométrico de las raíces, para el diseño de controladores PID en sistemas de segundo orden.
2. Desarrolló una solución computacional integrando herramientas libres como GeoGebra, JavaScript y Python para automatizar el análisis y diseño de sistemas de control.
3. Implementó interfaces gráficas interactivas que permiten visualizar en tiempo real el comportamiento dinámico del sistema bajo diferentes configuraciones del controlador.
4. Validó los resultados obtenidos con herramientas profesionales (MATLAB y SISOTOOL), demostrando la equivalencia y viabilidad de soluciones en software libre.
5. Comunicó de forma clara y estructurada el desarrollo técnico del proyecto, empleando lenguaje académico y técnico adecuado, en coherencia con los estándares del trabajo de grado.

12. SUGERENCIAS PARA TRABAJOS FUTUROS

Como proyección de este proyecto, se plantean ampliar las capacidades de la herramienta creada en GeoGebra en los siguientes aspectos:

- La integración de controladores no lineales: En la actualidad, los diseños se sustentan en controladores lineales del tipo PID en su mayoría; sin embargo agregar algoritmos como los controladores difusos o adaptativos podría resolver problemas en sistemas que presentan comportamientos no lineales o altamente variables y así ampliar la aplicación del entorno para procesos más complicados y reales.
- Incorporación de plantas de orden superior: La herramienta se centra principalmente en sistemas de segundo orden en su versión actual; estos son ampliamente utilizados en la enseñanza de control. Llevar su enfoque hacia las plantas de tercer orden o superiores, incluso incluyendo aquellas que presentan ceros en el numerador, podría facilitar la verificación de controladores en una variedad más amplia de modelos, y así aumentar su aplicabilidad en situaciones reales.
- La inclusión de perturbaciones externas y ruido en la simulación sería útil para probar la eficiencia y fiabilidad de los controladores en situaciones prácticas dentro del ámbito industrial; esto resultaría esencial para una evaluación más precisa y realista del rendimiento del sistema implementado.
- La herramienta actual se enfoca en sistemas SISO (Single Input Single Output), por lo que al ampliar su funcionalidad para incluir sistemas multivariados, los cuales podrán analizar de manera conjunta múltiples entradas y salidas. Este avance representa un paso significativo en el modelado y control de procesos industriales más complejos.
- En futuras actualizaciones de GeoGebra, se espera que se establezca de forma permanente la comunicación entre JavaScript y el entorno CAS (Sistema de Álgebra Computacional), mediante funciones como *evaluaComandoCAS()*. Esto facilita la realización de cálculos simbólicos directamente en el frontend sin requerir un backend adicional; simplificando así la arquitectura y mejorando el rendimiento global de la herramienta.
- Una interfaz más fácil de usar y adecuada para los alumnos podría consistir en la creación de plantillas visuales que simplifiquen la utilización de la herramienta en entornos educativos, facilitando así el uso de GeoGebra como plataforma para enseñanza de control automático en el ámbito universitario.

Estas mejoras harán que la propuesta sea más sólidamente considerada tanto en el ámbito académico como durante la fase de desarrollo de prototipos para soluciones de control industrial.

13. REFERENCIAS

- [1] J. C. E. Francisco Javier Córdoba Gómez, “Desarrollo y uso didáctico de GeoGebra,” CONFERENCIA LATINOAMERICANA COLOMBIA 2012 Y XVII ENCUENTRO DEPARTAMENTAL DE MATEMÁTICAS, 2012
- [2] R. Akkaya and Ü. Çakıroğlu, “The effects of GeoGebra on students’ mathematical skills: A meta-analysis study,” *IndoMath: Indonesia Mathematics Education*, vol. 7, no. 1, pp. 45–66, 2024. doi: 10.30738/indomath.v7i1.140.
- [3] M. W. Sunismi, “Meta-Analysis Study: Effectiveness of Using GeoGebra on Students’ Mathematical Ability,” *Jurnal Pendidikan Matematika Raflesia*, vol. 07, no. 02, pp. 140-146, 2022. [Online]. Available: <https://www.scribd.com/document/846239219/140-546-1-PB>
- [4] Gobierno de Canarias, “Recursos digitales – Bachillerato – Matemáticas II,” *Gobierno de Canarias*, 2025. [En línea]. Disponible en: <https://www3.gobiernodecanarias.org/medusa/ecoescuela/recursosdigitales/category/04-bachillerato/04mtimii/page/2/>
- [5] P. web oficial GeoGebra, “¿qué es GeoGebra?” Disponible en: <https://www.geogebra.org/about?lang=es> [Consultado:17-julio-2024].
- [6] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 13th ed. Upper Saddle River, NJ, USA: Pearson, 2017.
- [7] Treasury Wine Estates, “Treasury Wine Estates launches world-first \$10m wine automation system in the Barossa Valley,” Barossa Australia, 2021. [Online]. Available: <https://barossa.org.au/treasury-wine-estates-launch-world-first-10m-wine-automation-system>
- [8] Enermax, “Laboratory Oven with PID Temperature Controller,” Enermax Technology, 2023. [Online]. Available: <https://www.enermax.my/constanced9e2b6a0>. [Accessed: Aug. 18, 2025].
- [9] GeoGebra, «Find GeoGebra Resources», GeoGebra Help Center, [Online]. Available: <https://help.geogebra.org/hc/en-us/articles/8823429816221-Find-GeoGebra-Resources> [Accessed: 18-Aug-2025].
- [10] LovTechnology , ¿Qué es un Controlador PID (Controlador Proporcional-Integral-Derivativo): cómo funciona y para qué se utiliza? Disponible en: <https://lovtechnology.com/que-es-un-controlador-pid-controlador-proporcional-integral-derivativo-como-funciona-y-para-que-se-utiliza/> [Consultado:31-julio-2024]
- [11] L. S. del Río, “Los mil y un aportes de GeoGebra al estudio de la geometría tridimensional,” *Épsilon- Revista de Educación Matemática*, 2019.
- [12] T. GeoGebra, “Coming soon! create more powerful and dynamic visuals with python com- mands in GeoGebra,” Disponible en: <https://twitter.com/geogebra/status/1638586521769975813> [Consultado:27-julio-2024].
- [13] TIOBE the software quality company, TIOBE Index for July 2024 , Disponible en :

<https://www.tiobe.com/tiobe-index/> [Consultado:31-julio-2024]

- [14] O. Semenikhina , V. Proshkin , M. Drushlyak ,Automation of mathematical knowledge control within dynamic mathematics programs , Scientific Editor Eugenia Smyrnova-Trybulska, 2019
- [15] “PID control of inverted pendulum”. GeoGebra. [En línea]. Disponible: <https://www.geogebra.org/m/exhz9aqp> [Consultado: 10-agosto-2024]
- [16] “PID Regulator”. GeoGebra. [En línea]. Disponible: <https://www.geogebra.org/m/bFUvqfgm> [Consultado: 10-agosto-2024]
- [17] L. Benkhellat & D. Bensoussan, GeoGebra as a tool of design of ultrafast and robust controller, École de Technologie Supérieure, Montréal, Canada, 2017
- [18] Análisis visual de los resultados del diseño del sistema de control PID usando GeoGebra, vol. 2. Programa de estudios de ingeniería eléctrica de la Universidad Panca Budi, Medan, Indonesia: Rev. Ing. Tecnol. Marítimas, 2023.
- [19] Ramírez, J. M., & García, E. E. R. (27 de septiembre de 2007). SISTEMAS DE CONTROL I. Santiago de Cali-Colombia: Universidad del Valle
- [20] Sistemas de segundo Orden, Disponible en: <https://controlautomaticoeducacion.com/control-realimentado/sistemas-de-segundo-orden/> [Consultado: 21-noviembre-2024]
- [21] E. Tacconi, R. Mantz, J. Solsona, P. Puleston, Controladores Basados en Estrategias PID, LEICI Facultad de Ingeniería, UNLP, 2005
- [22] “Introducción a Guiones GeoGebraScript - GeoGebra Manual”. GeoGebra Manual. Accedido el 12 de agosto de 2024. [En línea]. Disponible: https://wiki.geogebra.org/es/Introducción_a_Guiones_GeoGebraScrip
- [23] “Control Tutorials for MATLAB and Simulink - Motor Speed: PID Control,” *University of Michigan: Control Tutorials for MATLAB and Simulink*, [Online]. Available: <https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed§ion=ControlPID>. [Accessed: 28-May-2025].
- [24] O. A. Cuya Solari, E. I. Flores Bustinza y J. E. Torres Chávez, “*Diseño de un Controlador PID Difuso para una planta de Segundo Orden Sub-amortiguada usando LABVIEW*,” *IEEE-UNAC*, [Online]. Available: <https://es.scribd.com/doc/142121381/Diseno-de-un-Controlador-PID-Difuso-para-una-planta-de-Segundo-Orden-Sub-amortiguada-usando-LABVIEW>. [Accessed: 28-May-2025].

ANEXOS

Link con instructivo para la descarga de los archivos de código para ambos métodos:

https://drive.google.com/file/d/1Gc7xjfe9cbXEQYLUNJckjR_9cXtteZ_l/view?usp=drive_link