UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR DE POSICIÓN DE MOTOR SIN ESCOBILLAS MEDIANTE CONTROL DE CAMPO ORIENTADO CON SINTONIZACIÓN POR ENJAMBRE DE PARTÍCULAS Y EVOLUCIÓN DIFERENCIAL

PRESENTADO POR:

BR. DIEGO ADRIANO CCORIMANYA CALLE

PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

ASESOR:

DR. ROGER JESUS COAQUIRA CASTILLO

CUSCO - PERÚ

2026

Universidad Nacional de San Antonio Abad del Cusco

INFORME DE ORIGINALIDAD

El que suscribe, asesor Roger Jesus Coaquira Castillo, quien aplica el software de detección de similitud al trabajo de tesis titulada: **"DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR DE POSICIÓN DE MOTOR SIN ESCOBILLAS MEDIANTE CONTROL DE CAMPO ORIENTADO CON SINTONIZACIÓN POR ENJAMBRE DE PARTÍCULAS Y EVOLUCIÓN DIFERENCIAL"**, presentado por: **DIEGO ADRIANO CCORIMANYA CALLE** con DNI Nro: **70420568**, para optar el título profesional de: **INGENIERO ELECTRONICO.**

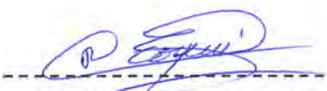
Informo que el trabajo de investigación ha sido sometido a revisión por dos veces, mediante el Software de detección de similitud, conforme al Art. 6° del *Reglamento para Uso del Sistema de Detección de Similitud de la UNSAAC* y de la evaluación de originalidad se tiene un porcentaje de 9% de similitud.

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No sobrepasa el porcentaje aceptado de similitud.	X
Del 11 al 30 %	Devolver al usuario para las subsanaciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe a l inmediato jerárquico, conforme al reglamento, quien a su vez eleva el informe al Vicerrectorado de Investigación para que tome las acciones correspondientes; sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de **asesor**, firmo el presente informe en señal de conformidad y adjunto la primera página del reporte del Sistema de Detección de Similitud.

Cusco, 11 de agosto de 2026



Dr. Roger Jesús Coaquira Castillo
Nro. de DNI 01333608
ORCID del Asesor: 0000-0003-3791-110X

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio:
   https://unsaac.turnitin.com/viewer/submissions/oid:27259:613643612?locale=es-MX

Diego Adriano Ccorimanya Calle

Tesis_repositorio.pdf

Universidad Nacional San Antonio Abad del Cusco

Detalles del documento

Identificador de la entrega
trn:oid:::27259:613643612

Fecha de entrega
10 ago 2026, 11:26 p.m. GMT-5

Fecha de descarga
10 ago 2026, 11:34 p.m. GMT-5

Nombre del archivo
Tesis_repositorio.pdf

Tamaño del archivo
16.4 MB

164 páginas

37.615 palabras

191.481 caracteres

9% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado

Fuentes principales

6%	Fuentes de Internet
3%	Publicaciones
6%	Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alerta de integridad para revisión

Caracteres reemplazados
68 caracteres sospechosos en N.º de páginas
Las letras son intercambiadas por caracteres similares de otro alfabeto.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Dedicatoria

Dedicado a mi familia, por su amor, paciencia y por brindarme la fortaleza necesaria para culminar este proyecto. Este logro es fruto de su constante aliento.

Agradecimientos

A Dios, por guiar mi camino y permitirme alcanzar esta meta. A mis padres, con todo mi corazón, pues este trabajo es fruto de su sacrificio y fe hacia mí; gracias por enseñarme con su ejemplo el valor de la responsabilidad y la honestidad, ya que a ustedes les debo todo lo que soy. A mi hermano y familia, por ser el soporte fundamental y brindarme el aliento necesario para transformar cada avance en una celebración compartida.

De manera especial, expreso mi profundo agradecimiento a la Universidad Nacional de San Antonio Abad del Cusco (UNSAAC), por ser mi casa de estudios y brindarme las herramientas necesarias para mi formación profesional. Asimismo, manifiesto mi gratitud al Dr. Roger Jesus Coaquira Castillo, por su guía, su tiempo y la confianza depositada en este proyecto; sus conocimientos y dirección técnica fueron fundamentales para la culminación de esta investigación.

Finalmente, a mis amigos y compañeros de estudio, por las jornadas compartidas, su amistad sincera y por hacer de esta etapa universitaria una experiencia memorable. A la vida, por rodearme de personas maravillosas cuyo apoyo hizo posible este logro. Este éxito es de todos.

Resumen

El uso de los motores sin escobillas en aplicaciones como vehículos eléctricos de baja potencia o sistemas industriales tiene especial relevancia debido a sus ventajas, como la eliminación de fricción de escobillas, una mayor densidad de potencia y una mejor eficiencia en comparación con motores DC convencionales. Estas características justifican la búsqueda de estrategias de control y métodos de optimización más eficientes y robustos.

En este contexto, la presente investigación trata del diseño e implementación de un controlador de posición para motores sin escobillas, siendo una variable que abarca de manera intrínseca el control de velocidad y torque interno del motor. Así, constituye una aplicación más completa y demandante, que integra los lazos internos de corriente y velocidad.

Para la sintonización se emplearon dos algoritmos estocásticos de optimización, el de enjambre de partículas (PSO) y el de evolución diferencial (DE). Implementados en tiempo real sobre una plataforma basada en una placa de microcontrolador STM32F405. Para limitar el espacio de búsqueda de los coeficientes del controlador, se usó una aproximación lineal del sistema y un análisis de estabilidad de dicha aproximación.

Finalmente, ambos algoritmos lograron un desempeño competitivo. En promedio, se redujo el sobreimpulso a menos del 0.6 % y el error en estado estacionario a menos del 0.04 %. La diferencia fue el número de generaciones que tardaron en llegar a su mejor valor, siendo más efectivo el PSO, con 21 iteraciones. Finalmente se evidencia la viabilidad de la optimización, y mejor rendimiento del PSO.

Palabras clave: Control de campo orientado (FOC), Motor sin escobillas, Optimización por enjambre de partículas (PSO), Optimización por evolución diferencial (DE).

Abstract

The use of brushless motors in applications such as low-power electric vehicles and industrial systems has gained special relevance due to their advantages, such as the elimination of brush friction, higher power density, and improved efficiency compared to conventional DC motors. These characteristics justify the search for more efficient and robust control strategies and optimization methods.

In this context, the present research addresses the design and implementation of a position controller for brushless motors, being a variable that intrinsically encompasses the control of motor speed and internal torque. Thus, it constitutes a more complete and demanding application, integrating the internal current and speed loops.

For the tuning process, two stochastic optimization algorithms were employed: particle swarm optimization (PSO) and differential evolution (DE). These were implemented in real time on a platform based on an STM32F405 microcontroller board. To limit the search space of the controller coefficients, a linear approximation of the system and a stability analysis of this approximation were used.

Finally, both algorithms achieved competitive performance. On average, the overshoot was reduced to less than 0.6 % and the steady-state error to less than 0.04 %. The difference was the number of generations required to reach their best value, with PSO being more effective, requiring 21 iterations. Finally, the feasibility of optimization and the improved performance of PSO are demonstrated.

Keywords: Field Oriented Control (FOC), brushless motor, Particle Swarm Optimization (PSO), Differential Evolution (DE).

Índice general

Índice de tablas

Índice de figuras

CAPÍTULO 1:

GENERALIDADES

1.1. Descripción de la realidad problemática

En el ámbito de la industria moderna, el control preciso de la posición es esencial para garantizar el rendimiento óptimo de los actuadores. Los motores sin escobillas son ampliamente utilizados en aplicaciones de este tipo debido a su alta eficiencia, tamaño compacto, diseño simple, alta densidad de potencia y una gran relación torque-inercia; lo cual los hace muy útiles en aplicaciones de servos o actuadores (Djouadi et al., 2024). Para controlar eficazmente este tipo de motores, uno de los métodos más usados es el control de campo orientado (FOC), también conocido como control vectorial (Mohanraj et al., 2022). Sin embargo, a pesar de sus ventajas, la implementación práctica del FOC enfrenta desafíos significativos, especialmente en lo que se refiere a la calibración precisa y eficiente de los controladores.

En sistemas robóticos, la precisión en el control de posición es fundamental. Actuadores mal calibrados pueden provocar errores en la ejecución de tareas, afectando la precisión y repetibilidad del robot. Por otro lado, en el área de la robótica industrial, los fallos llevan a complicaciones de seguridad y aumentos de costos en reparaciones por accidentes. Estas características también son particularmente críticas en áreas que influyen en la salud como la cirugía robótica; y en el área háptica, donde se aproxima la experiencia sensorial de los médicos cirujanos mediante actuadores robóticos que proveen retroalimentación (Adela, 2023).

Los métodos tradicionales de calibración de los controladores FOC suelen ser manuales y laboriosos además de requerir una inversión significativa de tiempo y recursos como queda evidenciado en los métodos de calibración de distintas marcas de dispositivos (ODrive Ro-

botics, 2025). Estas limitaciones pueden resultar en una calibración subóptima, lo que no solo afecta la precisión del control de posición, sino que también tiene implicaciones directas en la eficiencia energética del sistema. Un motor mal calibrado consume más energía de la necesaria, genera más calor y experimenta un desgaste prematuro, aumentando los costos operativos; además, se reduce la vida útil del equipo. Otra consecuencia se da en aplicaciones donde la eficiencia energética es crítica, como en robots móviles alimentados por baterías; en estas aplicaciones, una calibración deficiente puede limitar significativamente la autonomía y el rendimiento general del sistema.

Para evitar el trabajo y las ineficiencias del método tradicional iterativo, es posible emplear algoritmos de optimización tales como el algoritmo de Optimización por Enjambre de Partículas (PSO), que se basa en la exploración del espacio de posibles soluciones y está inspirado en la naturaleza, o el de evolución diferencial (DE); este pertenece a los algoritmos evolutivos, los cuales nacieron basados en el proceso natural de la evolución. Estos algoritmos funcionan automatizando la búsqueda de los parámetros de los controladores de manera independiente al número de variables a optimizar, pero no determinan los límites de búsqueda, puesto que normalmente se usan rangos arbitrarios en las investigaciones relacionadas que no se basaron en criterio alguno, como muestran Tien et al. (2021) y Abdelgar et al. (2023), donde se emplearon rangos arbitrarios los cuales se repitieron para cada coeficiente optimizado sin distinción. Esta falta de criterio al delimitar los rangos puede llevar a búsquedas muy extensas o a que no se logre una optimización efectiva.

1.2. Formulación del problema

1.2.1. Problema general

¿Cómo diseñar e implementar un controlador de posición para motores sin escobillas de corriente continua, en el cual se puedan optimizar los parámetros del controlador mediante algoritmos de PSO y algoritmos de evolución diferencial?

1.2.2. Problemas específicos

1. ¿Qué criterios de optimización y comparación aplicar a la calibración de los coeficientes de los controladores, que sean compatibles con el sistema?

2. ¿Cómo escoger el motor y circuitos de control físicos que serán usados para construir la planta y qué características deben tener para que sirvan a este propósito?

3. ¿Cómo comprobar la validez y efectividad de los métodos de optimización PSO y de evolución diferencial para el problema de optimización de parámetros del controlador?

1.3. Justificación de la investigación

1.3.1. Justificación social

La importancia de la investigación para la sociedad va de la mano con sus aplicaciones. Si bien se explora la aplicación del control de posición con un control FOC, esta se encuentra estrechamente relacionada con el control de velocidad o torque. Todos estos elementos, en conjunto, sirven para aplicaciones de vehículos eléctricos de media potencia (bicicleta o scooter), o la creación de dispositivos que van desde brazos robóticos hasta robots tipo SCARA, los cuales son muy efectivos con actuadores de motores sin escobillas por su alta velocidad y precisión durante el control. Este tipo de aplicaciones y su exploración buscan impulsar a la sociedad a realizar investigación en nuevas tecnologías, así como a desarrollar productos de calidad, como vehículos eléctricos de nicho. Al existir múltiples entornos particulares para realizar pruebas, esto tiene el potencial de beneficiar a la sociedad en su conjunto.

1.3.2. Justificación tecnológica

El estudio desarrollado constituye un aporte al conocimiento en el área de control de motores sin escobillas, siendo restringido a motores que tienen una fuerza contraelectromotriz

de forma sinusoidal, y busca comparar la optimización del funcionamiento del motor mediante algoritmos PSO y de evolución diferencial. Esto sirve para aplicar el conocimiento y los resultados en futuros modelos que busquen controlar motores en ambientes más hostiles y con variaciones de carga; ya que, teniendo como referencia el comportamiento y métodos de calibración en condiciones comunes, se pueden realizar generalizaciones para condiciones específicas.

Esta investigación, además, se constituye como el primer paso para implementaciones combinadas del FOC con otros tipos de controladores que reemplacen los controladores en lazos internos comunes PI por otros, tales como sistemas fuzzy o redes neuronales. Si bien en aplicaciones no muy exigentes no son necesarios, para aplicaciones altamente variables (como robots con carga variable o motores que trabajen en regímenes muy exigentes de cambios de temperatura) sí son útiles para mantener un funcionamiento adecuado y eficiente.

1.3.3. Justificación ambiental

La parte del proyecto que influye en el impacto ambiental es el ahorro de energía. Si bien los métodos de control de motores sin escobillas que incluyen realimentación, como el control FOC, son más complicados de implementar y requieren componentes de mayor costo (además de elementos adicionales), la mejora en la producción y variedad de estos componentes los hizo más accesibles con el tiempo. Así, las máquinas que utilicen este tipo de controladores, al igual que los vehículos eléctricos que emplean estos algoritmos, tienen ventajas en eficiencia frente a aquellos que usen métodos de control comunes de lazo abierto. Por consiguiente, esto implica un ahorro considerable de energía consumida, lo cual impacta positivamente en el medio ambiente.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar e implementar un controlador de posición para motores sin escobillas de corriente continua, el cual optimice los parámetros del controlador mediante algoritmos de PSO y algoritmos de evolución diferencial.

1.4.2. Objetivos específicos

1. Diseñar criterios de optimización para los coeficientes de los controladores basados en características del sistema.
2. Implementar una planta que emule un servosistema para el control de un motor sin escobillas de corriente continua (BLDC), determinando las características y especificaciones de los componentes adecuados que la componen.
3. Validar las técnicas de optimización seleccionadas y comprobar cuál es más efectiva mediante resultados que permitan su comparación.

1.5. Delimitación del estudio

1.5.1. Delimitación espacial

El estudio se llevó a cabo en la ciudad del Cusco, Perú, a 3328 msnm.

1.5.2. Delimitación temporal

El estudio se llevó a cabo entre los años 2025-2026.

CAPÍTULO 2:

MARCO TEÓRICO

2.1. Antecedentes

En el ámbito internacional se tienen investigaciones como la de **Optimal FOC-PID Parameters of BLDC Motor System Control Using Parallel PM-PSO Optimization Technique** presentada por Tien et al. (2021). En esta investigación se busca encontrar los parámetros óptimos de control para un sistema FOC-PID que controla un motor BLDC. Para este propósito, se utilizaron los métodos de optimización por enjambre de partículas común y una variación de este método llamada "MULTI-POPULATION TECHNIQUE" o técnica multipoblacional, en la que se divide la región de búsqueda de parámetros en muchas subsecciones de búsqueda. En cada una de estas secciones se utilizó el algoritmo de optimización por enjambre de partículas. Luego, el mejor resultado fue seleccionado usando la estrategia del mínimo posible. Este trabajo realizó pruebas de estos dos algoritmos mediante una simulación en Matlab. Los resultados muestran que la técnica multipoblacional tiene resultados competitivos con respecto a la técnica estándar, con valores de máximo sobrepico y tiempo de establecimiento prácticamente indistinguibles entre los resultados y una ventaja de un 0.2 % en el error de estado estacionario para la técnica multipoblacional. Por otro lado, esta técnica presenta una notable ventaja en el tiempo de simulación, rebajando los tiempos de prueba para un test de 10 generaciones y 24 partículas de 979 s a 308 s.

En el contexto internacional reciente se tiene el artículo **An intelligent optimization algorithm with novel fitness function for high-performance PMSM FOC** artículo propuesto por Zhou et al. (2025). En este artículo se exploró y probó la optimización del control vectorial (FOC) mediante un método de optimización inteligente que busca superar los retos

de rechazo de perturbaciones, adaptabilidad limitada del sistema y rendimiento en tiempo real, aplicado a un motor síncrono de imanes permanentes de 15 kW. El método propuesto consistió en usar un observador de estados mejorado y un algoritmo de optimización inteligente para sumar una señal a los controles PI estándar de la estrategia FOC y, al mismo tiempo, cambiar sus parámetros internos para lograr mitigar las variaciones del sistema general que se tienen ante variaciones en la resistencia e inductancia del motor usado. Las pruebas de efectividad se realizaron tomando el comportamiento del motor en condiciones de arranque, condiciones dinámicas y condiciones de estado estacionario. Los resultados mostraron que con estos métodos se tiene una mejora sustancial en pruebas dinámicas con golpes de carga y condiciones de estado estacionario; del mismo modo, en la prueba desde condiciones de arranque se logró una mejora en el tiempo de establecimiento de 0.132 s a 0.0882 s, lo que representa el 33 %.

Con referencia al control de posición de motores sin escobillas se tiene el artículo **Design and testing of a low-cost mobile platform for contactless building surveying** realizado por Jinlong et al. (2024). Este artículo trata del diseño y las pruebas de una plataforma para medir la distancia entre la línea de 1 metro de alto de un edificio sobre el nivel del suelo y el borde inferior de los interruptores eléctricos; la cual, según el trabajo, se considera una métrica crucial para evaluar la calidad de la construcción. Durante este proceso, se tomaron dos ángulos de un láser de medida; estos ángulos corresponden a los ángulos de posición de un motor sin escobillas de corriente continua que sujeta el láser. Estos ángulos fueron cruciales para los cálculos de las distancias y se controlaron mediante un control FOC que tuvo como apoyo un filtro Kalman, el cual se implementó para evitar el uso de sensores de posición. En el estudio se presentó una aplicación del controlador FOC para regular la posición del rotor de un motor. Los resultados demostraron una precisión aceptable al registrar, en las mediciones generales, un rango de error de 2 mm; dicho valor se encontró dentro del estándar industrial permitido para medidas de esta naturaleza, según lo expuesto por el autor.

Con referencia a las aplicaciones de este tipo de control se tiene el artículo **An elbow exoskeleton for haptic feedback made with a direct drive hobby motor** propuesto por Kim and Asbeck (2020). En este artículo se trata el diseño del codo de un exoesqueleto,

el cual fue usado para aplicaciones hápticas, además de su respectivo control de posición dentro de los ángulos de movimiento posibles de un codo humano. Para el movimiento, se utilizó un motor sin escobillas T-MotorMN7005-KV115 controlado por una tarjeta de control C2000TMS320F28069M de Texas Instruments. En el artículo se usó un método de estimación de ángulo de posición del rotor a través de un pin índice predefinido; por último, el controlador aplicado perteneció al tipo de control de campo orientado (FOC), cuyas ganancias P y D fueron calibradas de forma manual o empírica, basándose en el ancho de banda de funcionamiento.Los resultados preliminares obtenidos mostraron el potencial de un exoesqueleto para ser usado como herramienta de entrenamiento y guía de movimientos, además de proponer mejoras a futuro que están relacionadas con la sensación del usuario con respecto a las fuerzas de guía del sistema.

Sobre el control de ángulo en motores BLDC se tiene el artículo **Digital Position Control System With a BLDC Motor Using Field Oriented Control** propuesto por Lajić and Matić (2023). Este artículo tiene como objetivo presentar un sistema de control de posición para motores BLDC basado en un control FOC, que fue probado en una simulación en MATLAB. El artículo realizó un análisis del modelo matemático del motor, relacionando principalmente el torque con la corriente en las fases, para luego convertir el modelo al sistema de coordenadas (dq) mediante las transformadas de Clarke y Park, y así aplicar un controlador. Este controlador se sintonizó realizando un análisis matemático de los modelos del motor. Se inició por el modelo que representa el comportamiento de la corriente, el cual se usó para sintonizar los dos controladores PI del lazo de control de corriente; luego se usó el modelo aproximado del lazo de velocidad para sintonizar los componentes del controlador PI de este mismo lazo. Durante el cálculo de los valores, se tomaron en cuenta aproximaciones para partes del proceso de control como el inversor y el sensor de posición, que fueron aproximados a funciones de transferencia de primer orden. Luego de analizar los modelos matemáticos, el criterio usado para escoger los valores fue evitar completamente el sobreimpulso en el ángulo de posición. Este objetivo se logró obteniendo un sobreimpulso insignificante, además de un tiempo de estabilización de aproximadamente 1.8 s para una variación de ángulo de rotor de 100 grados y resistencia a los cambios de torque.

Con referencia a algoritmos de implementación de controladores FOC se tiene el trabajo **SimpleFOC: A Field Oriented Control (FOC) Library for Controlling Brushless Direct Current (BLDC) and Stepper Motors** desarrollado por Skuric et al. (2022). Este artículo trata de la implementación de una librería de control de posición, torque y velocidad de motores BLDC. La librería está orientada a proveer una implementación más genérica y fácil de usar del método FOC, para impulsar el desarrollo rápido de sistemas robóticos o configuraciones de control altamente dinámicas. El método para desarrollar esta librería es un desarrollo modular, enfocado en cada lazo de control y agregando filtros pasa bajos para las variables de corriente, velocidad y posición. Primero se desarrolló el lazo de control para la corriente, luego el de velocidad y, finalmente, el de posición. Cada uno de estos se puede usar de manera aislada y con diferentes variedades de encoders, desde magnéticos hasta incrementales.

En el campo de optimización se tiene el artículo **Evolutionary algorithms and their applications to engineering problems** presentado por Slowik and Kwasnicka (2020). En este artículo se muestra la familia de algoritmos evolutivos y para qué son utilizados, referenciando ejemplos de sus usos. Primero, se mostró a qué clase de algoritmos pertenecen como familia, siendo estos: The Genetic Algorithm, Genetic Programming, Differential Evolution, The Evolution Strategy y Evolutionary Programming. Estos algoritmos se encuentran clasificados entre los algoritmos de optimización que usan una búsqueda aleatoria. De entre todos estos algoritmos, el usado principalmente para la optimización dentro de un espacio continuo de búsqueda es el de Evolución Diferencial. Este trabajo además muestra en qué áreas de la investigación son aplicados, destacando la automatización en sistemas de control y la ingeniería eléctrica o electrónica. Finalmente, se concluye que los algoritmos son efectivos y se aplican en numerosos campos, incluyendo la automatización, donde se enfrentan los problemas de optimización computacional y la definición de objetivos de optimización claros para poder sacar provecho de los mismos.

Sobre los algoritmos de optimización basados en enjambres se tiene **An Overview of Variants and Advancements of PSO Algorithm** desarrollado por Jain et al. (2022). Este trabajo presentó una visión general de los algoritmos de enjambres de partículas (PSO),

sus conceptos básicos, parámetros, análisis teóricos, extensiones y probables hibridaciones. Estos algoritmos están basados en el estudio de comportamientos colectivos en enjambres de animales en la naturaleza y la forma de interacción entre sí sin necesidad de supervisión externa; además, en su conjunto tienen el objetivo de buscar una solución de calidad para sus respectivas aplicaciones usando el menor esfuerzo computacional posible. Finalmente, se concluye que los algoritmos de enjambres son populares entre la comunidad científica debido a su simplicidad, facilidad de implementación y competencia a la hora de resolver una gran variedad de problemas. Por otra parte, se resaltan sus múltiples variaciones e investigaciones para lograr mejores resultados con ellos durante las últimas décadas, lo que muestra que existe mucho potencial en el algoritmo para la optimización de procesos.

En el campo de algoritmos de optimizacion y las funciones de costo aplicadas se tiene **Optimized Luenberger Observer-Based PMSM Sensorless Control by PSO** desarrollado por Luo et al. (2022). Este trabajo presentó el control de velocidad de un motor PMSM mediante un controlador FOC, un observador de estados de Luenberger (LO) para la estimación de la velocidad y de las características del motor, y una optimización basada en el algoritmo PSO, empleando únicamente la función de penalización ITAE. Para corroborar la efectividad de la propuesta, se comparó el desempeño del sistema optimizado con el de un sistema que utilizó únicamente el observador de estados sin optimización. Los resultados mostraron que el sistema sin optimizar presentó un sobreimpulso del 18.8 % y un tiempo transitorio de 43 ms, mientras que el sistema optimizado alcanzó un tiempo de establecimiento de 11 ms y un sobreimpulso de 5.3 %, evidenciando una mejora significativa en el desempeño dinámico del sistema al reducir tanto el sobreimpulso como el tiempo de respuesta. Finalmente, en una planta física únicamente se realizó una validación experimental de los resultados obtenidos en simulación; no se llevó a cabo la optimización del sistema en tiempo real. Para ello, se implementaron la configuración optimizada y los coeficientes calculados en una placa STM32F030 como microcontrolador principal, corroborándose un comportamiento robusto frente a pequeños disturbios e incertidumbres en los componentes, lo que confirmó la viabilidad de la estrategia de control propuesta y la concordancia entre los resultados de simulación y los obtenidos en la implementación experimental.

2.2. Bases teóricas

2.2.1. Motor sin escobillas modelo matemático

Para esta investigación, se controla un motor sin escobillas (BLDC) que posee una fuerza contraelectromotriz sinusoidal. Este tipo de motor conserva las características de un motor de CC pero elimina el conmutador y las escobillas, debido a los problemas generales de los motores de CC comunes (Sahdev (2018)):

- Necesidad de reemplazo de escobillas de manera periódica.
- Tener velocidades limitadas por los problemas de conmutación.

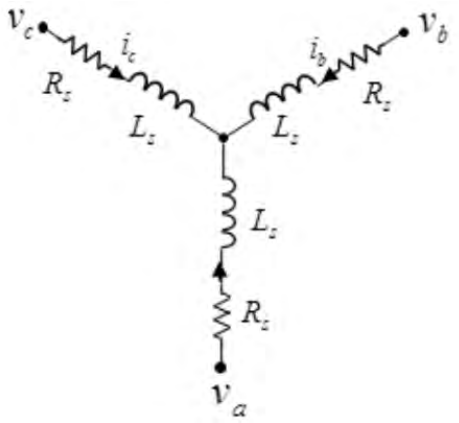
Este tipo de motores es alimentado mediante corriente continua y posee tres fases en la mayoría de los casos. Si bien este tipo de motores también cae en la clasificación de motor síncrono de imanes permanentes (PMSM) al tener una fuerza contraelectromotriz sinusoidal, se optó por esta denominación ya que el sistema está alimentado por corriente DC y las señales que salen del elemento final de control son de tipo PWM; además, se utilizaron motores de ambas denominaciones en los trabajos de referencia debido a su alta similitud en modelos y a que los motores BLDC también pueden tener fuerza contraelectromotriz sinusoidal (Matevosyan (2021)).

Un motor trifásico tiene tres fases independientes y un rotor con imanes permanentes; primero se modelan las corrientes inducidas en el estator debido a las entradas de voltaje (Mohanraj et al. (2022)). Como parte de las bases, el modelo de circuito pasivo del motor se muestra en la figura 2.1. Este modelo permite representar el comportamiento eléctrico de cada una de las fases del estator y analizar la interacción entre las variables eléctricas y mecánicas del sistema. Debido a que se trata de un sistema trifásico balanceado, esta representación facilita la aplicación de las transformadas de Clarke y Park, simplificando el análisis y el desarrollo de estrategias de control. A partir de esta representación es posible establecer las ecuaciones que describen la dinámica del motor, considerando los efectos de la resistencia, la inductancia y la fuerza contraelectromotriz generada por el rotor. Dicho mode-

lado constituye la base para el diseño de estrategias de control avanzadas, ya que proporciona una descripción matemática del sistema sobre la cual pueden implementarse algoritmos de estimación y control.

Figura 2.1

Circuito eléctrico interno del motor sin escobillas



Nota. *Adaptado de MATLAB (2025b)*

A este modelo circuital se le agregan fuentes de voltaje en cada una de las fases, estas representan la fuerza contraelectromotriz generada durante la rotación del motor y son representadas con la *e* en cada fase.

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \frac{d}{dt} \left(\begin{bmatrix} L_{aa} & L_{ab} & L_{ac} \\ L_{ba} & L_{bb} & L_{bc} \\ L_{ca} & L_{cb} & L_{cc} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \right) + \begin{bmatrix} e_a \\ e_b \\ e_c \end{bmatrix} \tag{2.1}$$

- V_x, V_y, V_z representan los voltajes de las fases del motor.
- I_a, I_b, I_c representan las corrientes del estator.
- L_{aa}, L_{bb}, L_{cc} representan las inductancias del estator.
- e_a, e_b, e_c Representan la fuerza contra electromotriz.
- R Representa la resistencia de fase, asumiendo resistencias iguales y que no hay cambio en la reluctancia entre el rotor y estator con la posición angular.

Como se aprecia en la figura 2.1, la conexión de las fases del motor es de tipo estrella; por lo tanto, las corrientes internas están relacionadas.

$$i_a + i_b + i_c = 0 \tag{2.2}$$

De la ecuación anterior se deriva la relación.

$$Mi_b + Mi_c = -Mi_a \tag{2.3}$$

Que es de ayuda al simplificar las matrices con la forma.

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \frac{d}{dt} \left(\begin{bmatrix} L-M & 0 & 0 \\ 0 & L-M & 0 \\ 0 & 0 & L-M \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \right) + \begin{bmatrix} e_a \\ e_b \\ e_c \end{bmatrix} \tag{2.4}$$

La fuerza contraelectromotriz puede tener distintas formas según el tipo de motor; en nuestro caso, se usó un motor con forma de onda sinusoidal, la cual se representa en la siguiente ecuación en el término f.

$$\begin{bmatrix} e_a \\ e_b \\ e_c \end{bmatrix} = \omega_m \lambda_m \begin{bmatrix} f_{as}(\theta_r) \\ f_{bs}(\theta_r) \\ f_{cs}(\theta_r) \end{bmatrix} \tag{2.5}$$

- ω_m es la velocidad de rotación del motor.
- λ es el flujo de acoplamiento del motor.
- f_{as} es el flujo magnético del motor y representa la fuerza contraelectromotriz en diferentes posiciones o instantes distintos.

Por otro lado, el torque se calcula con la siguiente ecuación, la cual deriva de la ecuación de potencia mecánica de un motor eléctrico.

$$T_e = \frac{1}{w_m}(e_a i_a + e_b i_b + e_c i_c) \tag{2.6}$$

Agregando el componente del torque generado por el motor se muestra la ecuación de movimiento del sistema.

$$T_e - T_m = J\frac{dw}{dt} + Fw(t) \tag{2.7}$$

- J es la inercia combinada.
- F es el coeficiente de fricción mecánica.

La velocidad angular del rotor y la de los campos eléctricos del sistema están relacionadas de la siguiente manera.

$$\omega_m = P(\omega_r) = P\frac{d\theta_r}{dt} \tag{2.8}$$

- P es el número de pares de polos del motor.
- θ_r es el angulo mecánico del rotor.
- ω_r es la velocidad mecánica del rotor.
- ω_m es la velocidad angular de la onda contraelectromotriz sinusoidal.

Estas relaciones están en un contexto de tres fases, si se cambia el sistema de referencia a las coordenadas d y q mediante las transformadas Clarke y Park se tienen las mismas relaciones en diferentes planos de referencia, obteniendo las siguientes ecuaciones (MATLAB (2025b)).

$$v_d = R_s i_d + L_d\frac{di_d}{dt} - P\omega_m i_q L_q \tag{2.9}$$

$$v_q = R_s i_q + L_q\frac{di_q}{dt} + P\omega_m(i_d L_d + \lambda_m) \tag{2.10}$$

$$v_0 = R_s i_0 + L_0\frac{di_0}{dt} \tag{2.11}$$

$$T = \frac{3}{2}N\left(i_q(i_d L_d + \lambda_m) - i_d i_q L_q\right) \tag{2.12}$$

Estos son los modelos fundamentales en ambos planos de referencia, y contribuyen a realizar las simulaciones en MATLAB para hacer validaciones previas del método de sintonización planteado.

- R_s es la resistencia de fase.
- L_d es la inductancia de fase en el plano d.
- L_q es la inductancia de fase en el plano q.

2.2.2. Transformada Park y Clarke

Para realizar el control del motor sin escobillas, se necesita controlar las corrientes del modelo mediante el voltaje aplicado, debido a que realizar el control de tres salidas de voltaje en el sistema de coordenadas estacionarias (abc) es complicado; para el control FOC, se opta por transformar el sistema anterior a un marco de referencia giratorio ortogonal (dq). Este proceso se realiza mediante las transformadas de Clarke y Park, que convierten los componentes del marco (abc) a dos componentes de un marco ortogonal ($\alpha\beta$) que es estacionario, para luego usar la transformada de Park, que convierte el marco en un marco giratorio ortogonal (dq).

Transformada Clarke

Es una transformación algebraica que permite convertir un sistema trifásico (abc) a un sistema de dos fases de coordenadas ($\alpha\beta$). El origen del sistema trifásico es el eje "a" y el del sistema de dos fases es el eje α . Esta transformación es válida para los fasores de corriente y voltaje que rigen el comportamiento de los motores trifásicos.

Para tres fasores separados por 120 grados sexagesimales que representen ondas de voltaje o corriente, los cuales se transforman en 2 fasores separados por 90 grados, se comprueba la transformada mediante las siguientes ecuaciones.

$$U_a = 1; \quad U_b = e^{j\left(\frac{2\pi}{3}\right)}U_a; \quad U_c = e^{j\left(-\frac{2\pi}{3}\right)}O_a; \quad U_a + U_b + U_c = 0 \tag{2.13}$$

$$U_\alpha = 1; \quad U_\beta = jU_a \tag{2.14}$$

Al darle un valor unitario a las magnitudes de los fasores anteriores se obtiene la equivalencia que debe ser cumplida, donde *C* representa la matriz de la transformada de Clarke.

$$\begin{bmatrix} U_\alpha \\ U_\beta \\ 0 \end{bmatrix} = C \begin{bmatrix} U_a \\ U_b \\ U_c \end{bmatrix} \tag{2.15}$$

$$\begin{bmatrix} 1 \\ e^{j\left(-\frac{2\pi}{3}\right)} \\ e^{j\left(\frac{2\pi}{3}\right)} \end{bmatrix} = C \begin{bmatrix} 1 \\ j \\ 0 \end{bmatrix} \tag{2.16}$$

Despejando los valores se obtienen los valores que debe tomar la matriz C para que ocurra la transformada.

$$C = K \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.17}$$

Con esta transformación se convierten los componentes como se muestra en las figuras 2.2 y 2.3. El componente de magnitud K se incluye con el fin de mantener la amplitud pico de las componentes o la potencia instantánea, por lo que toma los siguientes valores.

$$K_{amplitud} = \frac{2}{3} \tag{2.18}$$

$$K_{potencia} = \frac{\sqrt{2}}{3} \tag{2.19}$$

La transformada inversa representa la inversa de la matriz utilizada como transformada, respetando el coeficiente de magnitud respectivo.

$$C^{-1} = \frac{1}{K}\begin{bmatrix} \frac{2}{3} & 0 & \frac{2}{3} \\ -\frac{1}{3} & \frac{\sqrt{3}}{3} & \frac{2}{3} \\ -\frac{1}{3} & \frac{\sqrt{3}}{3} & \frac{2}{3} \end{bmatrix} \tag{2.20}$$

Figura 2.2
Transformada de Clarke en gráficas en el tiempo

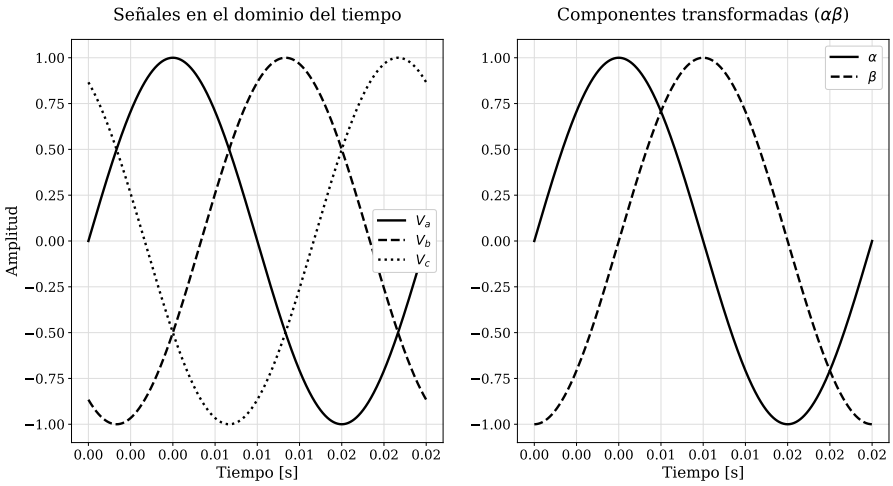
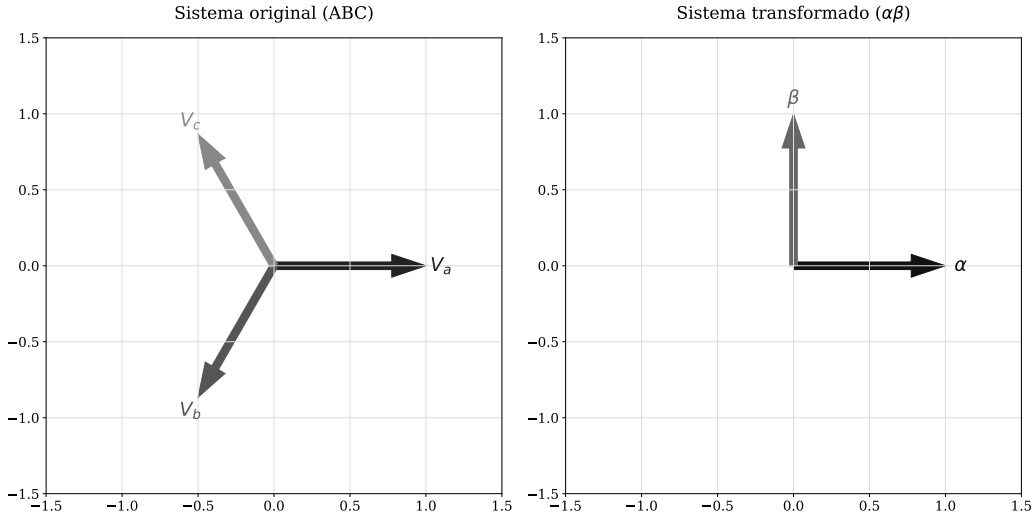


Figura 2.3
Transformada de Clarke en esquema fasorial



Transformada Park

Esta transformación algebraica convierte el sistema de dos ejes($\alpha\beta$) estático a un sistema de dos ejes rotatorio (dq). Este marco de referencia gira típicamente sincronizado con una

cantidad física de la máquina, como el vector de flujo del rotor o el vector de tensión del estator.

Debido a que el tipo de motor usado es síncrono, el vector de flujo gira con la misma frecuencia que el rotor, y al ser usado como referencia para las componentes de corriente o voltaje estos permanecen constantes en su posición angular, esta es la interpretación física de esta transformada la cual ayuda a simplificar aún más el desarrollo del control. La representación en matrices de esta transformada se da del siguiente modo, siempre y cuando se tome el sistema rotatorio alineado a la componente α.

$$\begin{bmatrix} U_d \\ U_q \\ 0 \end{bmatrix} = P \begin{bmatrix} U_\alpha \\ U_\beta \\ 0 \end{bmatrix} \tag{2.21}$$

$$\begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) & 0 \\ -\sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix} \tag{2.22}$$

Para cuando el sistema esté 90 grados adelantado al eje alpha se tiene la siguiente relación.

$$\begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} = \begin{bmatrix} \sin(\omega t) & -\cos(\omega t) & 0 \\ \cos(\omega t) & \sin(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix} \tag{2.23}$$

Las inversas respectivas para cada caso son:

$$\begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} = \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) & 0 \\ \sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix} \tag{2.24}$$

$$\begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} = \begin{bmatrix} \sin(\omega t) & \cos(\omega t) & 0 \\ -\cos(\omega t) & \sin(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix} \tag{2.25}$$

La representación gráfica de cómo la transformada cambia las señales en el tiempo y en un diagrama fasorial se muestra en las figuras 2.4 y 2.5

Figura 2.4
Transformada de Park en el tiempo

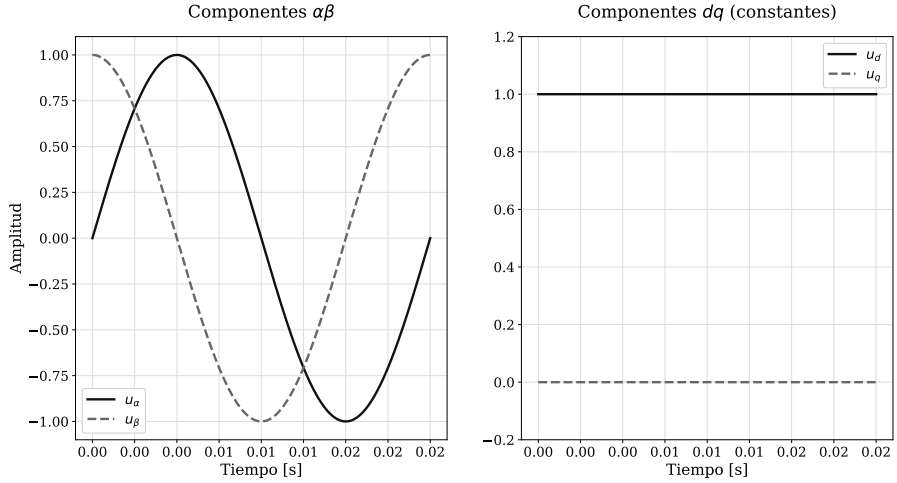
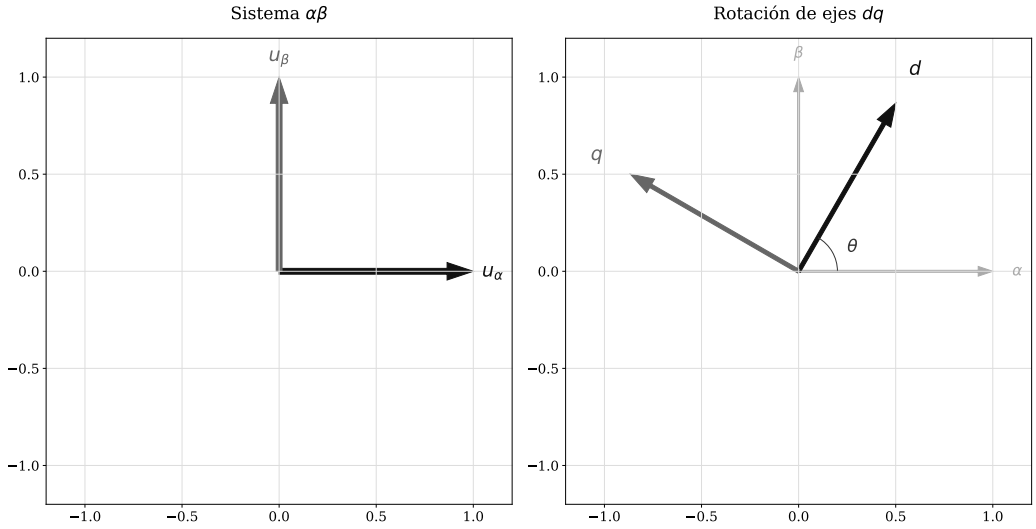


Figura 2.5
Transformada de Park en esquema fasorial



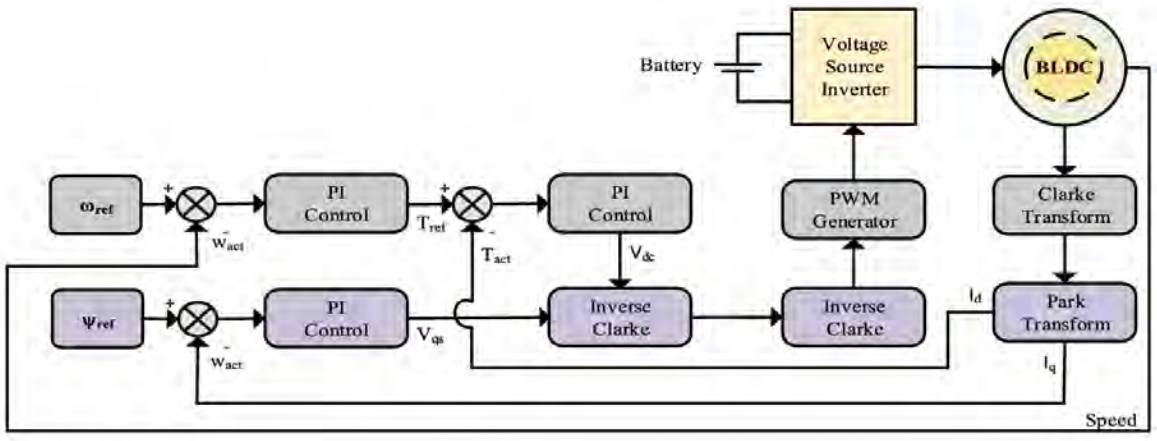
2.2.3. Controlador FOC esquema y funcionamiento

El control FOC es uno de los métodos más antiguos para motores de imanes permanentes que permite evitar la ondulación del par, manteniendo el ángulo entre el flujo de campo del rotor y estator con un valor de 90 grados (Mohanraj et al. (2022)). Este algoritmo se aplica a motores que funcionan en el dominio (dq) y tiene la ventaja de permitir el control de manera simultánea del ángulo de giro y el torque aplicado al rotor.

La estructura general de este tipo de controladores se puede dividir en dos capas de control principalmente. En la primera capa se encuentra el control de los valores de corriente i_d y i_q con el propósito de generar el vector de control deseado en el estator. Mientras tanto, en la segunda capa se controla la velocidad del rotor, la cual depende del valor i_q como se ve en el esquema del controlador en la figura 2.6.

El propósito del control de campo orientado (FOC) es regular las corrientes en el eje directo (i_d) y en el eje en cuadratura (i_q) para lograr el torque deseado. Al controlar el torque por separado, es posible optimizar la relación de Torque por Amperio (MPTA); esto a su vez optimiza el consumo de corriente, lo que tiene efectos positivos en la eficiencia del sistema (Chavan et al. (2023)).

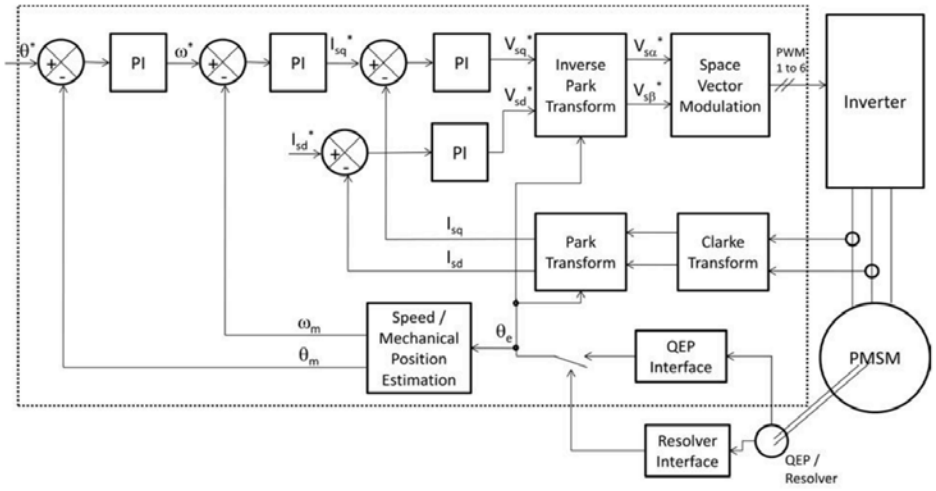
Figura 2.6
Diagrama genérico de controlador FOC



Nota. *Adaptado de Mohanraj et al. (2022)*

En el lazo más externo se controla la velocidad angular del motor mediante acciones sobre el componente i_q de corriente. Si bien en el modelo tradicional se encuentran solo estos lazos, y en trabajos como Tien et al. (2021), se utilizan solo estos lazos para controlar la posición intercambiando la realimentación de velocidad por una de posición angular, se aprecia que en otros trabajos se utilizó un lazo extra al de velocidad con un controlador proporcional para controlar la posición como en la figura 2.7, ya que la posición es directamente proporcional a la velocidad (Lajić and Matić (2023)). Según otras fuentes, este resultado se sustenta de manera experimental, donde el control P resultó suficiente para controlar el ángulo del sistema (Skuric et al. (2022)).

Figura 2.7
Diagrama de control de posicion de BLDC



Nota. *Adaptado de Texas Instruments (2021)*

2.2.4. Controlador PI

Este tipo de controladores tienen una acción de control definida por la siguiente ecuación que es dependiente del error de entrada (Ogata (2010)).

$$u(t) = K_p e(t) + \frac{K_p}{T_t} \int_0^t e(t)\, dt \tag{2.26}$$

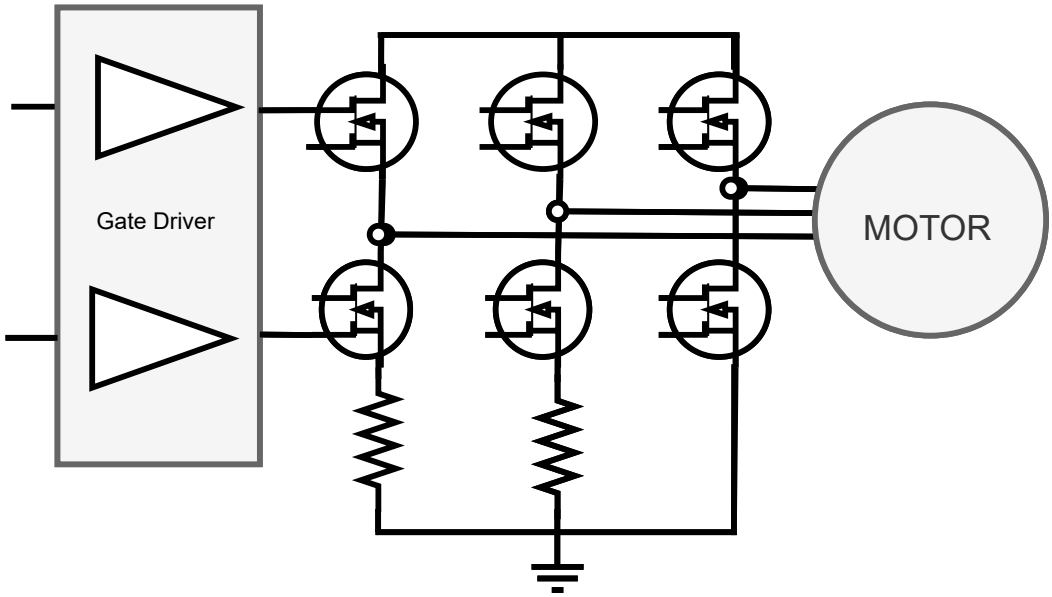
En el esquema básico y genérico del control FOC se aprecian claramente tres lazos de control. Un lazo para las corrientes i_d e i_q respectivamente, donde el torque del motor es

proporcional a la corriente en i_q, y otro para i_d que no influye en el torque, por lo tanto se tiene como práctica general igualarlo a cero. Estos lazos emplean controladores PI para regular el flujo de corriente y torque que depende de ellos, estos controladores deben responder más rápido que los lazos externos. Para estos lazos se opta por controladores PI en lugar de PID porque la acción derivativa amplifica el ruido de las mediciones ante cambios abruptos, una característica indeseable en sistemas que inherentemente presentan diversos ruidos parásitos. Por consiguiente, generalmente los controladores de estos lazos son de tipo PI.

2.2.5. Circuitos de control FOC o de control sin escobillas

En un controlador FOC se necesitan transistores como elementos finales de control, estos deben estar en una configuración de triple medio puente en H (Triple 1/2-H Bridge Driver), como se muestra en la figura 2.8.

Figura 2.8
Circuitos de triple medio puente H



Nota. *Adaptado de Texas Instruments (2021)*

Esta configuración permite el manejo de las tres fases y se puede encontrar en diversos circuitos integrados, como por ejemplo el DRV8313, que lo implementa en su interior. Los transistores comúnmente usados para estas configuraciones son MOSFET de potencia de canal N o IGBT. Para controlar esta configuración se necesitan seis señales de control que

llegan a la configuración, ya que, se tienen seis transistores que controlar. Para esto se aplican diferentes tipos de modulación específica como la SPWM (Modulación por Ancho de Pulso Sinusoidal), que es una modulación que busca generar ondas sinusoidales partiendo de señales digitales PWM.

2.2.6. Tipos de modulación

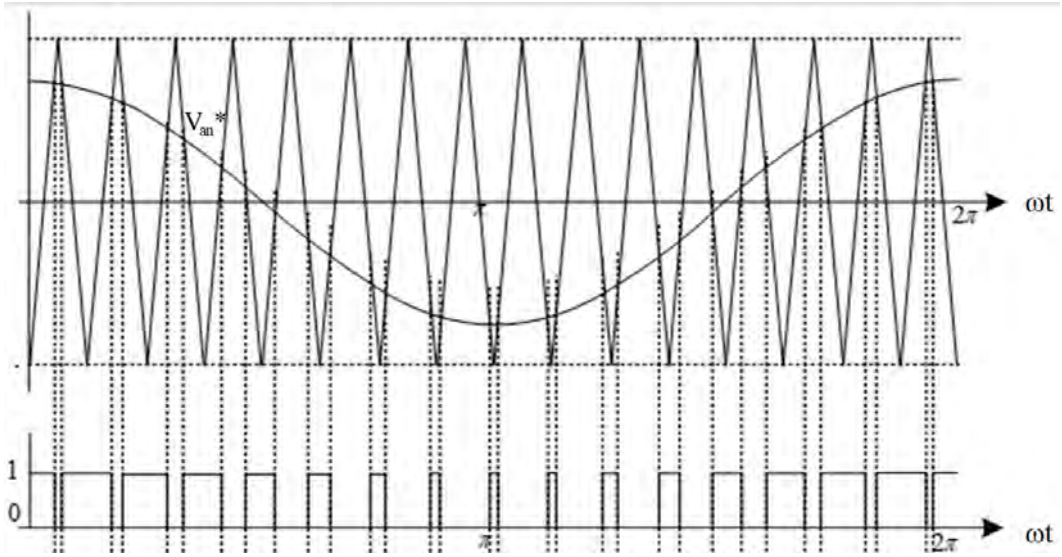
Los motores sin escobillas dependen intrínsecamente de los controladores electrónicos del voltaje en sus entradas, a diferencia de los motores DC comunes. Si no se energiza el sistema de forma precisa y coordinada, el sistema no presenta giro alguno. Para este propósito se emplean técnicas de modulación por ancho de pulso(PWM).

Modulación SPWM

La Modulación Sinusoidal por Ancho de Pulso (SPWM) es una técnica PWM que busca generar una tensión de salida cuya componente fundamental sea sinusoidal, de frecuencia y amplitud controladas. El principio básico consiste en comparar una señal de referencia sinusoidal de baja frecuencia (la frecuencia fundamental deseada) con una señal portadora de alta frecuencia en comparación relativa a la otra señal, que típicamente es una onda triangular. Este es el tipo de modulación más común, utilizado para generar salidas de voltaje trifásicas controladas (Park et al. (2023)).

Los puntos de intersección entre la onda de referencia sinusoidal y la onda portadora triangular determinan los instantes exactos en que los interruptores del inversor tienen que conmutar (realizar el encendido o apagado), tal como se muestra en la figura 2.9. Como resultado, se genera una señal PWM en la salida del inversor, la cual consiste en una serie de pulsos de amplitud constante (de valor igual a la tensión del bus DC) pero cuyo ancho varía a lo largo de su ciclo fundamental siguiendo una ley sinusoidal. Esta modulación permite obtener una forma de onda de salida con menor contenido armónico y un mejor aprovechamiento de la tensión disponible.

Figura 2.9
Modulación SPWM para una fase



Nota. *Adaptado de Tawfiq et al. (2024)*

Modulación SVPWM

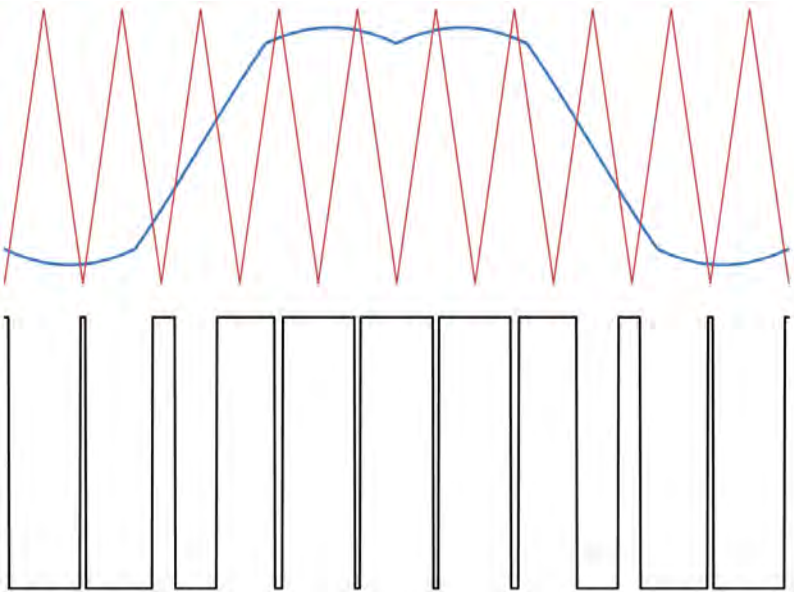
La modulación por vector espacial (SVM), también es conocida como modulación por ancho de pulsos de vector espacial (SVPWM). Esta técnica permite enviar ciclos de trabajo a los inversores del sistema, que en este caso son los transistores; los cuales, a su vez generan la tensión necesaria para activar el motor (MATLAB (2025a)).

SVPWM proporciona voltajes de línea de carga PWM que, en promedio, son iguales al voltaje sinusoidal de referencia. La forma de onda para la implementación de SVPWM se basa en la selección adecuada de los estados de conmutación del inversor, el cálculo de los períodos de conmutación apropiados y la transformación del vector espacial. SVPWM ofrece la reducción de las pérdidas de conmutación. Como resultado, la capacidad de salida del inversor mejora. SVPWM hace un uso más eficiente de la tensión de alimentación en comparación con SPWM y reduce el contenido armónico en las formas de onda de la tensión de salida (Tawfiq et al. (2024)). Como desventaja, se puede mencionar su mayor complicación durante la implementación.

Tomando solo una fase de modulación, la señal de voltaje generada mediante modulación por vectores espaciales (SVM) tiene una forma similar a la onda azul presentada en la figura

2.10. Esta señal es comparada con una onda triangular, representada en rojo en la misma figura. Cada vez que el valor de la onda triangular supera al de la señal SVM, se activa el interruptor inferior. El resultado de este proceso es una señal PWM (Modulación por Ancho de Pulso), que se emplea para regular el funcionamiento de la fase correspondiente del puente en H.

Figura 2.10
Modulación SVPWM



Nota. *Adaptado de Texas Instruments (2021)*

Conmutaciones para los métodos de modulación

En este tipo de modulaciones se busca generar tensiones específicas en los terminales del motor. Estas tensiones se pueden representar como vectores en el espacio que forman un hexágono. Dentro de este hexágono, los vectores básicos se obtienen mediante las conmutaciones de los interruptores ideales, en este caso representados como transistores, configurados como se muestra en la figura 2.11, y mediante combinaciones de estos vectores básicos se pueden generar vectores en cualquier posición y con cualquier magnitud, tal como se muestra en la figura 2.12.

Una vez definidos los vectores básicos, el algoritmo de modulación determina el sector del hexágono en el que se encuentra el vector de referencia deseado. A partir de esta información, se seleccionan los dos vectores activos adyacentes y los vectores nulos co-

rrespondientes para sintetizar, durante cada período de conmutación, un vector promedio equivalente al vector de referencia. La duración de aplicación de cada vector se calcula en función de la magnitud y el ángulo del vector de referencia, de manera que el valor medio de la tensión aplicada al motor reproduzca la referencia con un error mínimo. Por último, el orden específico de pares de activación se muestra en el cuadro de la figura 2.13.

Figura 2.11

Funcionamiento de los circuitos puente H

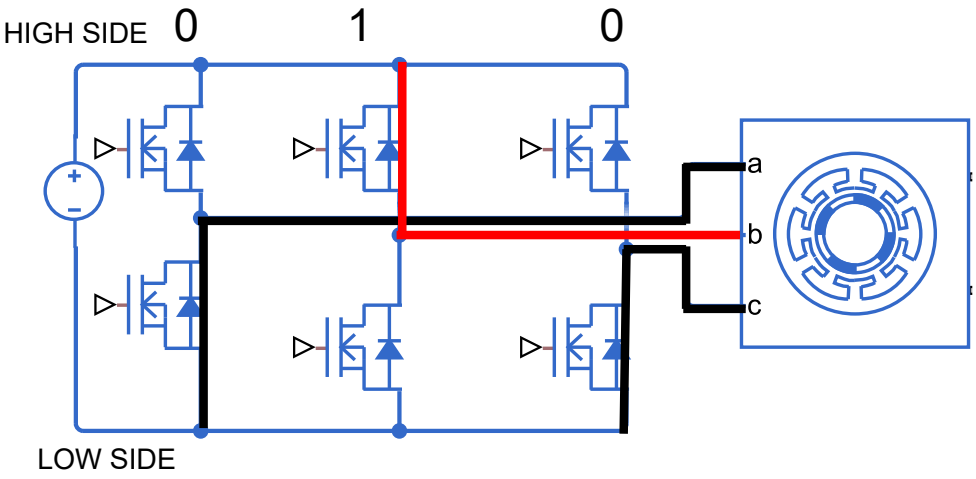
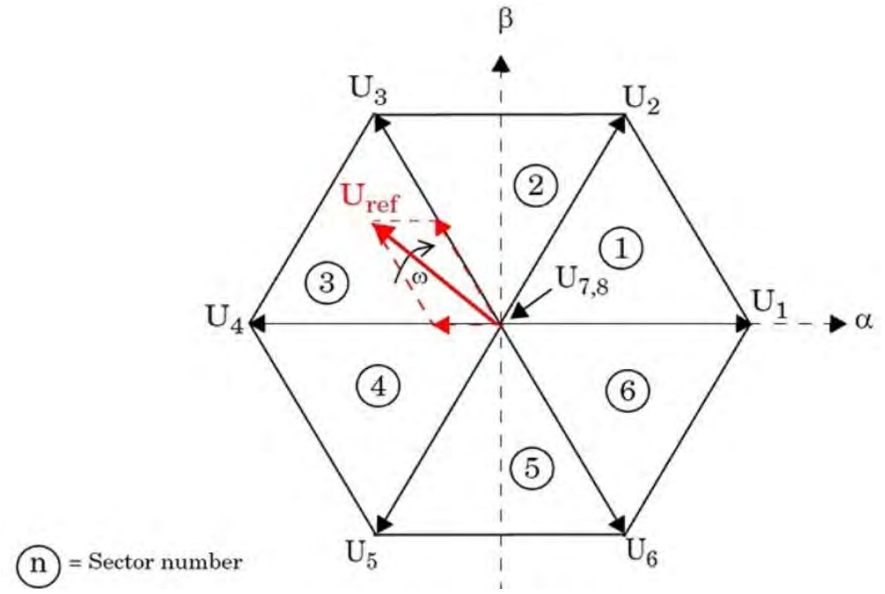


Figura 2.12

Vectores de voltaje formados por las conmutaciones



Nota. *Adaptado de MATLAB (2025a)*

Figura 2.13
Tabla de conmutaciones para los circuitos puente H

Space Vector	S1	S3	S5
U1	1	0	0
U2	1	1	0
U3	0	1	0
U4	0	1	1
U5	0	0	1
U6	1	0	1
U7	0	0	0
U8	1	1	1

Nota. *Adaptado de MATLAB (2025a)*

De manera ilustrativa, para la creación del vector U_{ref} se utiliza una secuencia de conmutación de los dos vectores adyacentes, que serán U_3 y U_4; dependiendo de cuál predomine, se determina el ángulo del vector. Luego, para generar diferentes magnitudes del vector, se mantiene esta conmutación durante un periodo de tiempo, y el resto del mismo se activa un vector nulo (U_7 o U_8), regulando así su magnitud. El objetivo final de este tipo de modulación es producir secuencias de conmutación que generen vectores de control de rotación continua y magnitud arbitraria, de acuerdo con las necesidades del controlador (MATLAB (2025a)).

2.2.7. Algoritmos de optimización

Optimización por enjambre de partículas(PSO):

La optimización por enjambre de partículas (PSO) es una técnica metaheurística que pertenece a los métodos de búsqueda basados en grupos o poblaciones de animales y se emplea para resolver problemas de optimización. Su funcionamiento se inspira en el comportamiento colectivo de las aves imitando cómo un grupo de aves se dirige hacia un objetivo (como una fuente de comida) utilizando tanto su propia experiencia como la información compartida por el resto del grupo. Cada partícula ajusta su posición basándose en su mejor experiencia propia y en la mejor posición encontrada por el enjambre completo, reorganizándose con-

tinuamente para formar una configuración óptima y automática. El objetivo del enjambre es conseguir el mejor estado o posición dentro de todo el rango de búsqueda limitado (Jain et al. (2022)). Este tipo de algoritmo trabaja en iteraciones generando enjambres de soluciones, donde cada partícula representa una solución potencial para el problema a optimizar.

La actualización de velocidad y posición del enjambre de partículas en este algoritmo es el principal mecanismo de movimiento y búsqueda. El término VK será la velocidad de la partícula. La velocidad de la partícula en una iteración (i+1) se actualiza según la siguiente ecuación:

$$VK_{(i)+1} = VK_{(i)} + c_i r_i (p_{best} - X_k(i)) + c_2 r_2 (g_{best,i} - X_k(i)) \tag{2.27}$$

- Vk es la velocidad de la partícula en cada iteración.
- Xk es la posición de la partícula.
- Pbest es la mejor posición individual encontrada.
- Gbest es la mejor posición encontrada por el enjambre.
- C1 y C2 son los coeficientes de la aceleración, que controlan la influencia de las mejores posiciones ya sea propia o global.
- r1 y r2 son números aleatorios distribuidos entre 0 y 1 que mantienen el nivel de diversidad durante las iteraciones.

Esto significa que el algoritmo tiene tres componentes para realizar la actualización de la velocidad (Tien et al. (2021)). Estos son:

- El momento, representado por la velocidad anterior del algoritmo, que puede ir acompañado de una ganancia W para alterar el peso que tiene en la velocidad y que, de ser el caso, se clasifica como PSO estándar (Jain et al. (2022)).
- La parte cognitiva, representada por el coeficiente c1 que recuerda los máximos anteriores locales.

- La parte de aprendizaje social, representada por el coeficiente c2 y relacionada con el máximo general.

Para realizar la actualización de la posición de las partículas, la posición de la partícula es dependiente íntegramente de la posición anterior y de la velocidad de la iteración actual.

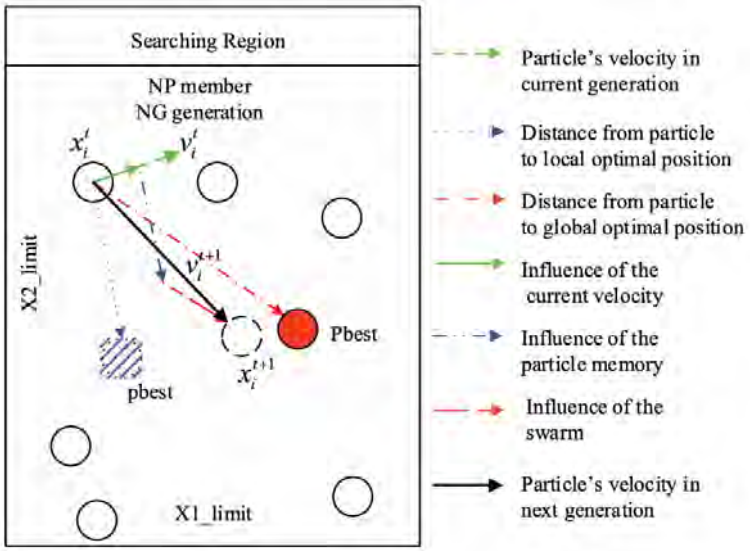
$$X_K(i+1) = X_K(i) + V_K(i+1) \tag{2.28}$$

- Xk es la posición de la partícula.
- Vk es la velocidad actual de la partícula.

Las influencias de cada componente se pueden apreciar en la figura 2.14, que muestra un vector por cada componente de la velocidad y la variación de posición de la partícula.

Figura 2.14

Visualización del funcionamiento del PSO



Nota. *Adaptado de Tien et al. (2021)*

Algoritmo de evolución diferencial(DE)

Este tipo de algoritmos se clasifica dentro de los algoritmos evolutivos y se usa principalmente para realizar optimizaciones, en casos donde los espacios de búsqueda son continuos. Este tipo de algoritmo tiene ventajas sobre los algoritmos evolutivos en situaciones donde se

tiene que lidiar con problemas extensos, ya que es computacionalmente menos complejo y tiene un uso eficiente de la memoria (Slowik and Kwasnicka (2020)).

En este tipo de algoritmo tiene tres procesos fundamentales:

1. La selección de la población.
2. El cruce de los elementos de la población.
3. La mutación posterior al cruce de elementos.

La selección se realiza de manera arbitraria, siempre y cuando los valores escogidos estén dentro del rango del mayor y menor valor permitidos. Luego de que se escogen los valores, se los evalúa con la función de ajuste, que es la función que se quiere optimizar o que proporciona los valores de desempeño de los elementos de la población actual.

Luego del proceso de selección, se procede a realizar la mutación. Esta se realiza usando elementos diferentes al elemento base a ser cruzado. Con estos tres elementos se realiza una mutación para variar sus características y obtener resultados diferentes.

Con el elemento ya mutado, se procede a realizar un cruce con el elemento objetivo original. De este proceso, finalmente se obtiene el elemento de prueba. Este elemento es evaluado mediante la función de ajuste y, si tiene un mejor desempeño que su predecesor, se toma como el nuevo elemento en esa posición.

Este proceso se lleva a cabo un número definido de veces o hasta que se cumplan los objetivos planteados como condiciones para la optimización. Por otro lado, si algún elemento resulta fuera del rango de prueba, debe restringirse al mismo usando el método que se vea conveniente.

El pseudocódigo estándar de este tipo de algoritmos se muestra en la figura 2.15 y en él se incluyen estos tres pasos fundamentales; además de procesos específicos extras, como la designación de funciones o coeficientes usados.

Figura 2.15
Pseudocódigo del algoritmo de evolución diferencial

Algorithm 2 Differential Evolution

```
determine objective function (OF)
assign number of generation to 0 (t=0)
randomly create individuals in initial population P(t)
while termination criterion is not satisfied do
    t=t+1
    for each i-th individual in the population P(t) do
        randomly generate three integer numbers:
        r1, r2, r3∈[1; population size], where r1≠r2≠r3≠i
        for each j-th gene in i-th individual (j∈[1; n]) do
            v_{i,j} = x_{r1,j} + F · (x_{r2,j} − x_{r3,j})
            randomly generate one real number rand_j ∈ [0; 1)
            if rand_j < CR then u_{i,j} := v_{i,j}
            else
                u_{i,j} := x_{i,j}
            end if
        end for
        if individual u_i is better than individual x_i then
            replace individual x_i by child u_i individual
        end if
    end for
end while
return the best individual in population P(t)
```

Nota. *Adaptado de Slowik and Kwasnicka (2020)*

2.3. Variables e indicadores

2.3.1. Variable independiente

- Voltaje de entrada al motor BLDC

2.3.2. Variables dependientes

- Posición del rotor del motor
- Corriente que circula por el motor

2.4. Definición de términos básicos

2.4.1. Voltaje de entrada al motor BLDC

El voltaje de entrada es aquel voltaje que se pone en las 3 fases que posee un motor trifásico, en nuestro caso es un voltaje DC que es controlado y enviado por un CI antes de esta etapa y por lo tanto viene en la forma de ondas PWM. Esta variable tiene como unidad de medida los voltios (V).

2.4.2. Posición del rotor del motor

Es la posición física del rotor del motor. Se puede medir mediante un encoder y es importante ya que es la variable que se busca controlar, por lo tanto de su precisión depende el éxito del modelo de control utilizado. La unidad de medida es el radian(rad) o los grados sexagesimales (°) a partir de un punto de referencia inicial.

2.4.3. Velocidad del rotor

Es la velocidad relacionada con el rotor, si bien se puede medir de manera directa mediante la onda de voltaje contraelectromotriz del motor este método tiene un ruido muy considerable, por lo cual se opta por su aproximación a partir de la medida de la posición. Las unidades para su cuantificación son radianes por segundo (rad/s) o revoluciones por minuto (RPM).

2.4.4. Corrientes de fase

Las corrientes de las fases se encuentran en el interior del motor, el cual tiene bobinas conectadas comúnmente en forma de estrella. Estas corrientes están relacionadas justamente por este tipo de configuración y de estas corrientes depende el torque que tiene el motor eléc-

trico, por lo cual es una variable de medida crucial. Esta variable se cuantifica en amperios (A).

2.4.5. Momento de inercia

Esta unidad depende de la posición de la masa con respecto al eje de rotación de un objeto y mide su resistencia a los cambios de velocidad angular. Esta se mide en ($kg \cdot m^2$).

CAPÍTULO 3:

METODOLOGÍA

3.1. Tipo de investigación

La presente investigación es de tipo aplicada, debido a que se aplican resultados y conocimientos de investigaciones previas para poder dar solución a la problemática de control y comparación de metodologías.

3.2. Alcance de la investigación

Dado que el objetivo de la investigación es implementar un controlador de posición para motores BLDC el cual optimice sus parámetros de lazos de control mediante algoritmos de PSO y algoritmos de evolución diferencial(DE), este tipo de trabajo es EXPLICATIVO puesto que se utilizaron técnicas ya conocidas, se realizaron comparaciones exponiendo los resultados y la tendencia de comportamientos del sistema; para esto se midió su comportamiento de manera cuantitativa con parámetros como el IAE o ITAE.

3.3. Diseño de la investigación

Se diseña la investigación como un diseño experimental, ya que se controló durante todos los experimentos la variable independiente para realizar el análisis del comportamiento en los resultados de la variable dependiente.

3.3.1. Grupo de control

Lo constituye el motor sin escobillas escogido, incluidas las cargas agregadas que afectan su movimiento y sin incluir el sistema de control de alimentación.

3.3.2. Preprueba

La preprueba de la investigación consistió en una prueba con el controlador aplicado pero con los coeficientes sin optimizar y valor unitario, con lo cual se tuvo un punto de partida a partir del cual se realizaron comparaciones; además de las comparaciones mutuas de los resultados de optimización con los algoritmos aplicados.

3.3.3. Postprueba

Se realiza la medición de características del comportamiento del control de movimiento luego de aplicados los algoritmos de sintonización. Se midió el tiempo de establecimiento, el error de estado estacionario y el sobreimpulso del movimiento del rotor.

3.4. Unidad de análisis

La investigación se centró en el control de posición del rotor de un motor sin escobillas.

3.5. Técnicas de recolección de datos

Para la recolección de datos se usó la técnica de observación. Esta se realizó tanto visualmente para comprobar la coherencia de comportamientos de los ángulos, como mediante el uso de sensores para la recolección de datos de corriente y posición; estos fueron integrados en las placas de control y se realizaron gráficas de sus resultados con respecto al tiempo; además, se almacenaron sus valores en archivos de datos de tipo texto.

3.6. Validez y confiabilidad de los instrumentos

La validez y confiabilidad de los sensores usados fue previamente puesta a prueba mediante el uso de multímetros digitales que sirvieron como patrón, con los cuales se corroboró el nivel de error que tienen las variables medidas. Por otro lado, para la recolección de ángulos de posición se realizó una corroboración visual mediante un transportador y sus ángulos de magnitud ya conocida.

3.7. Limitaciones de la investigación

- Las pruebas se realizaron únicamente usando el motor Eaglepower LA8308 KV90.
- No se realizó una caracterización completa del motor usado para la realización de las simulaciones previas a la implementación, puesto que solo se tomaron valores característicos fundamentales, como la inductancia y la resistencia interna.

3.8. Modelado del motor

En esta sección se desarrolla el modelado matemático del motor usado; este es un modelo básico de motores sin escobillas, cuyo motivo es servir de base para los valores iniciales de los coeficientes del controlador planteados en este trabajo. Con esta noción previa se evitó el uso de parámetros que lleven a la inestabilidad los cuales no tendría sentido probar y en aplicaciones reales pueden poner en riesgo los equipos y a las personas.

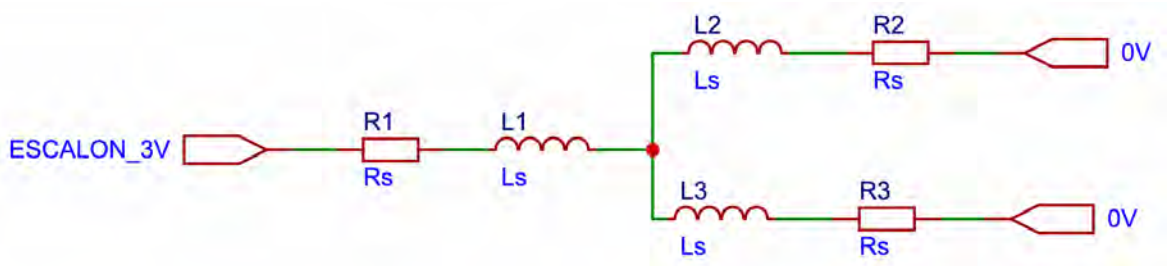
3.8.1. Resistencia del motor

La resistencia interna del motor se puede medir de dos formas, siendo una forma común su medición mediante multímetros o medidores RLC que tengan gran precisión, ya que la magnitud de la resistencia es reducida, llegando al orden de miliohmios (Infineon Technologies (2023)). Otra manera es usar el mismo controlador del motor que tiene sensores de

corriente integrados para realizar una medición indirecta mediante la ley de Ohm como se realizó en ODrive Robotics (2025). Se optó por este tipo de medición ya que en esta se midieron los valores del sistema incluyendo sus conexiones de funcionamiento, es decir que se incluyeron los valores adicionales de la placa de control que se agregaron a la resistencia interna del motor si estuviera desconectado; además de ser práctico debido al uso de los componentes incluidos en la placa de control.

La resistencia de fase es aquella que se encuentra entre una fase y el punto común de conexión de las fases; esta se puede medir con la configuración de la figura 3.1 si se asume que las 3 fases tienen una misma resistencia e inductancia. La resistencia fue hallada de los valores de voltaje aplicado y corriente medida a través de la fase 1.

Figura 3.1
Configuración del circuito para la medición de resistencia e inductancia.



Como se observa en la figura 3.1 se aplicó un escalón de voltaje de 3V por una de las fases y las otras dos se pusieron a un voltaje 0. Por lo tanto la resistencia de la fase obedece a la siguiente relación derivada del análisis del comportamiento del circuito para una entrada DC.

$$Rs = \frac{2}{3}\frac{V}{I_{R1}} \tag{3.1}$$

Se ejecutaron los siguientes pasos.

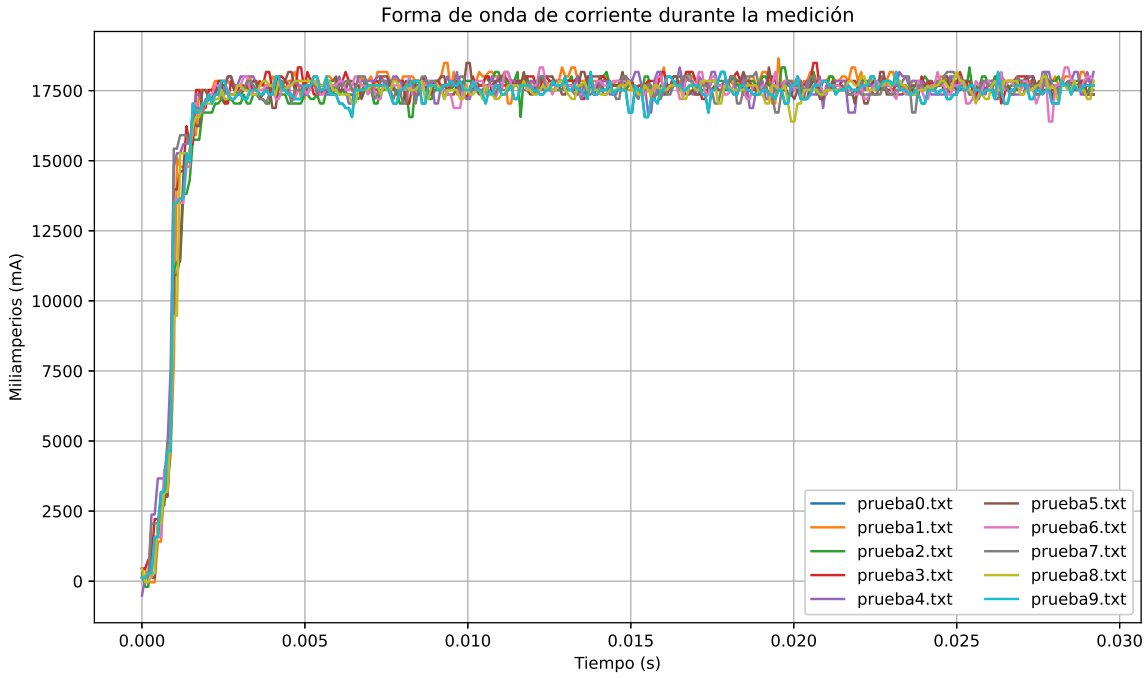
- Se envió un escalón de voltaje al sistema y se esperó a que llegara a la zona de estado estacionario.
- Se midió el promedio de 100 muestras de corriente cuando estuvo en estado estacio-

nario para evitar el ruido.

- Se usó la ley de Ohm y la relación de Rs para despejar la resistencia del voltaje y la corriente.

El gráfico de la figura 3.2 muestra la medición real de 10 pruebas realizadas, en las que se realizó el proceso antes mencionado.

Figura 3.2
Respuesta de la corriente ante una entrada escalón de 3 V



El resultado para la resistencia de fase fue el siguiente:

$$Rs = \frac{2}{3}\frac{3V}{17.96A} \tag{3.2}$$

$$Rs = 0.111\Omega \tag{3.3}$$

3.8.2. Inductancia del motor

Para realizar la medida de la inductancia se tienen las opciones del método directo con un medidor RLC e indirecto como en el caso de la resistencia. Se utilizó el método indirecto de manera complementaria al método que se utilizó para la medición de la resistencia. La

inductancia se obtuvo a partir del análisis de la respuesta transitoria de la corriente ante la aplicación de un escalón de tensión.

La inductancia de fase es aquella que hay entre una fase y el punto común de las bobinas. Si se asume que las 3 fases tienen la misma inductancia entonces la inductancia medida entre la fase uno y la unión de las fases 2 y 3 tiene un valor de 3/2 de la inductancia de fase, siguiendo el diagrama de la figura 3.1. Se siguió el siguiente procedimiento aprovechando el promedio de las pruebas de escalón de voltaje entre fases.

- Se envió un escalón de voltaje al sistema y se esperó a que llegara a la zona de estado estacionario.
- Se midió cuánto tarda el sistema en llegar al 63.2 % del valor final.
- Se halló el promedio de la constante de tiempo medida en múltiples pruebas para hallar una aproximación de la inductancia.

El tiempo que tarda un sistema en llegar al 63.2 % representa la constante de tiempo que tienen el sistema; debido a las alteraciones que tiene por el ruido del sistema se tomó el promedio de 10 medidas con lo cual se obtuvo un cálculo aproximado.

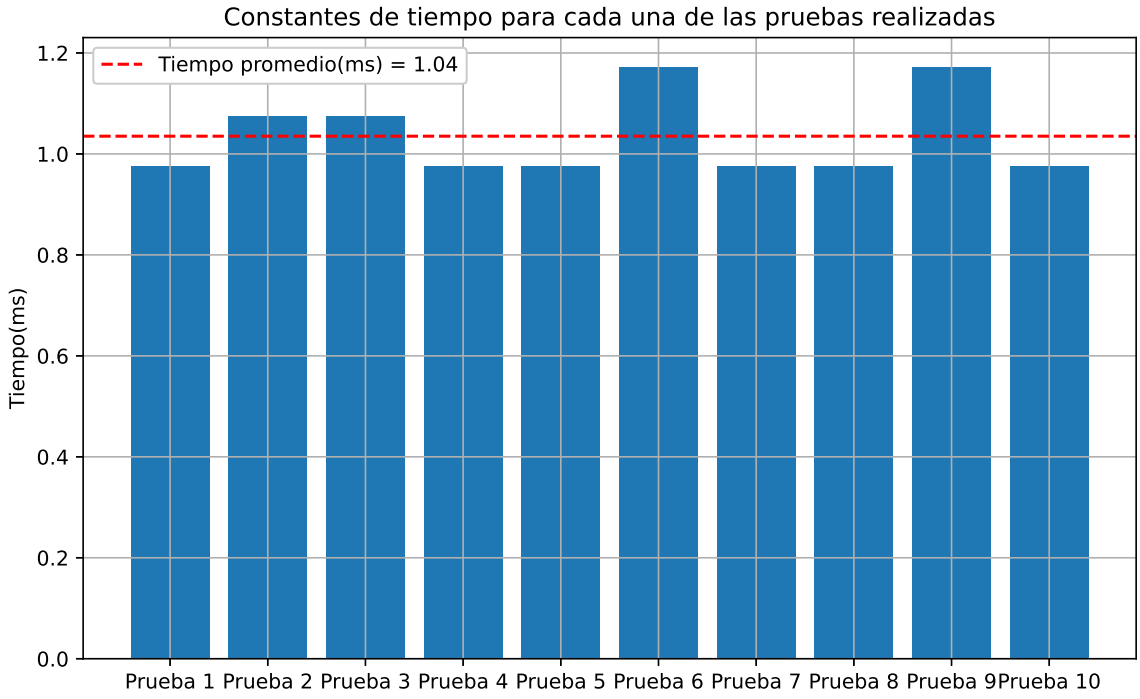
El gráfico de la figura 3.3 muestra el tiempo en muestras que le tomó a la prueba en escalón el llegar al 63 % de su valor final y el promedio que es de 10.6 muestras para una frecuencia de muestreo de 10240Hz. Con base en estos valores se pudo obtener la constante de tiempo del sistema y en base a ello la inductancia.

$$t = \frac{Ls}{Rs} \tag{3.4}$$

$$Ls = 1.04ms(111m\Omega) \tag{3.5}$$

$$Ls = 115.6uH \tag{3.6}$$

Figura 3.3
Promedio de inductancia de fase del motor



3.8.3. Fuerza contra electromotriz

Este factor fue importante para el tipo de control ya que indica fácilmente los límites físicos de velocidad a los que se puede desplazar el motor; de este modo, proporciona un límite basado en parámetros físicos para la velocidad que se puede alcanzar. En este caso se midió la fuerza contra electromotriz directamente del motor con un osciloscopio siguiendo el método propuesto por STMicroelectronics (2018).

El proceso fue el siguiente:

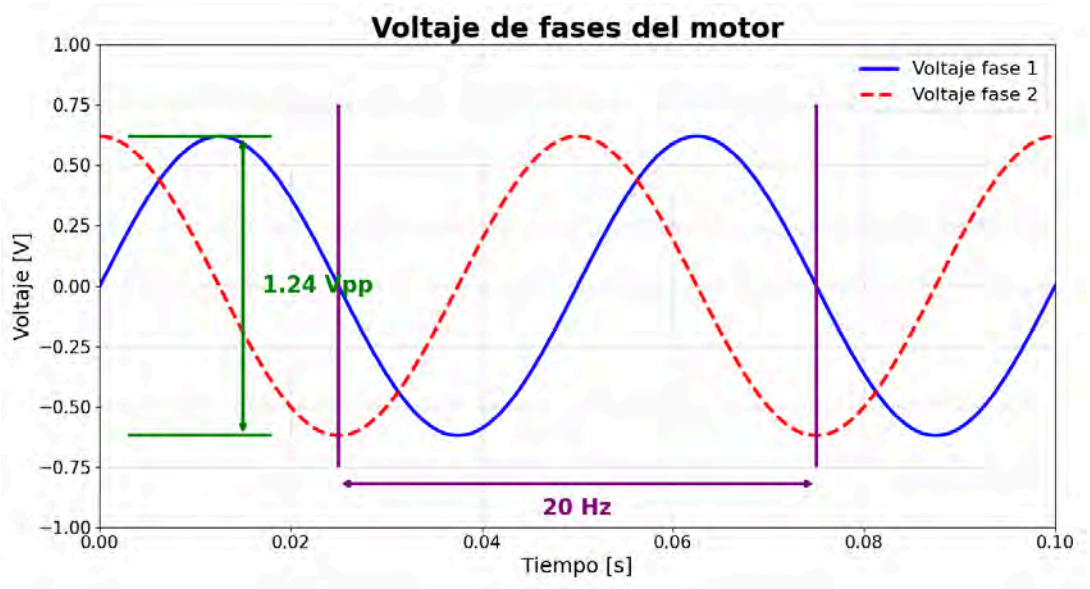
1. Se unió el motor a un mecanismo impulsor, que puede ser un taladro u otro motor; pero tienen que tener una velocidad de rotación estable.

2. Se midió el voltaje entre fases, el cual fue una onda sinusoidal y del cual se pudo calcular la onda contraelectromotriz entre fase y neutro.

En la figura 3.4 se muestra la onda contraelectromotriz para una rotación del motor impulsor de 1Hz, esta prueba se realizó para 1Hz y 2Hz; de estos valores se extrae el promedio

y se toma el dato correspondiente.

Figura 3.4
Gráfico de la fuerza contraelectromotriz 1Hz



Las relaciones obtenidas fueron:

$$\frac{V}{Hz} = \frac{1.24}{1} = 1.24 \tag{3.7}$$

$$\frac{V}{Hz} = \frac{2.5}{2} = 1.25 \tag{3.8}$$

A partir de estas pruebas, se obtuvo el voltaje pico a pico entre dos fases promedio por unidad en Hz.

$$V_{PicoPico} = 1.245V/Hz \tag{3.9}$$

Con base en esta medida, además fue posible calcular el valor de la fuerza contraelectromotriz que, según (MATLAB (2025b)), es el voltaje pico inducido por unidad de velocidad rotacional de cada fase del motor.

$$\frac{V_{Pico}}{Hz} = 1.245\frac{V}{Hz}\frac{1}{2}\frac{1}{\sqrt{3}} \tag{3.10}$$

Por lo tanto, el voltaje pico de fase por Hz de frecuencia de giro del rotor del motor será:

$$\frac{V_{Pico}}{Hz} = 0.359\frac{V}{Hz} \tag{3.11}$$

3.8.4. Momento de inercia de la carga

Esta magnitud sirvió de ayuda para tener una base en la calibración del lazo de velocidad y posición. Esta magnitud fue estimada puesto que no se tuvo una medición directa o de alta exactitud debido a la falta de algunos parámetros que la conforman, como la masa del rotor del motor usado. Sin embargo, al poseer la información de los componentes más representativos del momento de inercia de la planta de prueba se realizo una aproximación. Para aproximar el valor de este dato se utilizaron las ecuaciones que representan el momento de inercia de una barra, una carga y un disco con respecto a un eje de giro.

Para una masa aislada

$$I = ML^2 \tag{3.12}$$

Donde M es masa y L la distancia al centro de gravedad que sirve para objetos cuyo centro de masa es fácilmente localizable o definido. Luego para una barra ideal.

$$I = \frac{1}{12}ML^2 \tag{3.13}$$

Donde M es la masa de la barra y L la longitud total de la misma; por último, para un anillo circular.

$$I = \frac{1}{2}M(a^2 + b^2) \tag{3.14}$$

Siendo a y b los radios exterior e interior del anillo circular. Con estas tres ecuaciones se aproximó el momento de inercia de todas las cargas del rotor de la planta asumiendo que

fueran cargas ideales, puesto que no se tomó en cuenta el ancho de las barras, o la densidad no uniforme del disco que representa el rotor, las cargas fueron:

- 2 barras metálicas de 38.4 y 37.6 cm con pesos de 82.4 y 81 gramos respectivamente.
- 2 masas de 500 g con un centro de masa ubicado a 16.2cm del centro de giro.
- 2 soportes de las masas de 13.4 g cada uno ubicados a 17.2 cm del centro de giro.
- 1 soporte de rotor a barras impreso de 5.6 g, con forma cilíndrica, diámetro externo de 4 cm y diámetro interno de 1.5 cm.

La suma total y los resultados de los cálculos se muestran en la tabla 3.2.

Tabla 3.1
Cuadro de componentes del momento de inercia y suma total

Objeto	**Momento inercia** $Kg\ m^2$
Masas 500g	0.026244
Soportes de masa	0.000792
Barra de metal 1	0.000954
Barra de metal 2	0.00101
Soporte central	0.00000128
Total	0.029

3.8.5. Coeficiente de torque con respecto a la corriente

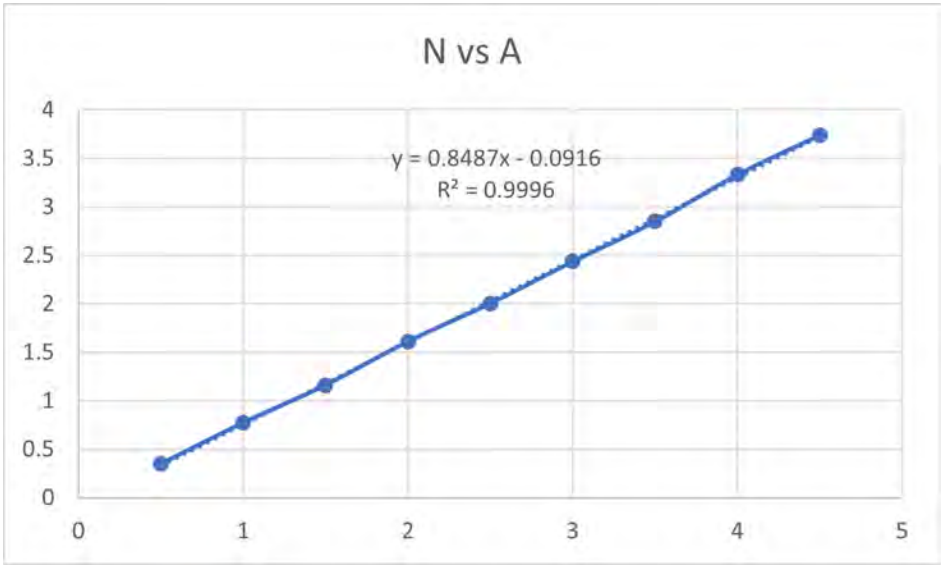
Esta magnitud depende de la construcción del motor usado y, en el caso de un motor sin escobillas, si se dispone de un controlador de corriente se puede realizar su medida de manera directa. Ya que el controlador usado también puede funcionar como controlador de corriente, se midió esta magnitud de este modo. Haciendo una analogía con los motores DC con escobillas, esta variable representa la magnitud de torque que se genera por cada amperio que se haga pasar por el devanado del estator.

$$T = K_t I \tag{3.15}$$

Esta magnitud es un coeficiente de relación entre variables idealmente lineal y, para el presente caso se lo hallo siguiendo el procedimiento descrito a continuación:.

- Se diseñó un brazo de 10cm para medir el torque en su extremo.
- Se utilizó una balanza para medir la fuerza que genera el motor por cada amperio de corriente y se convirtió la medida de la balanza a newtons.
- Se realizó la curva de correlación de los parámetros de donde se extrajo la magnitud.

Figura 3.5
Relación entre la fuerza en newtons medida en la balanza y la corriente del motor



Del ajuste lineal realizado se obtiene que la constante fue de valor $K_t = 0.0849\frac{Nm}{A}$ con una muy pequeña desviación, pero con un comportamiento prácticamente lineal al ser R^2 muy cercano a 1, como se muestra en la figura 3.5.

3.9. Calibración de parámetros y definición de intervalos de búsqueda para los algoritmos

En esta sección se describe cómo se hallaron los rangos de calibración para el caso de los algoritmos de optimización y cómo se calibró el lazo de corriente interno. Estos cálculos representaron una base donde ejecutar los algoritmos, puesto que si bien se pudieron

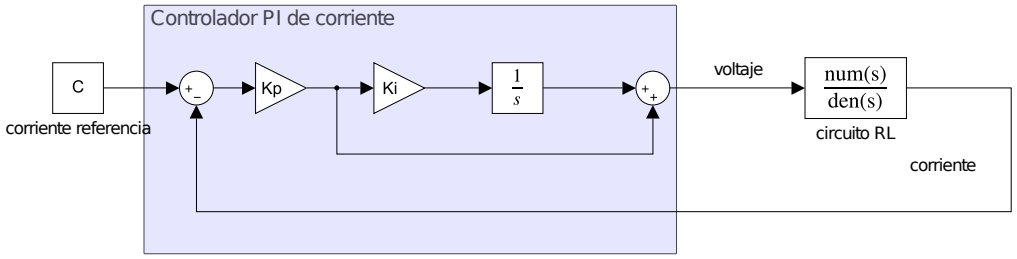
tomar valores arbitrarios extensos para asegurar que se encontrara una zona de estabilidad, la búsqueda se hubiera vuelto muy extensa; además, se habría tenido muchos valores que produjeran inestabilidad, lo que podría llegar a dañar el sistema.

3.9.1. Calibración de lazo de corriente:

La calibración del lazo de corriente se realizó siguiendo los pasos vistos en Wilson (2023) donde, en base al modelo básico de circuito RL del devanado del motor, se pudo lograr el control de su comportamiento incluyendo un control PI, lo cual a su vez permite escoger el ancho de banda del sistema. Esta calibración depende estrictamente de la precisión de la medición de los valores de R y L del circuito.

Según el gráfico de la figura 3.6 del lazo de control, se muestra la forma que tiene el controlador PI, esta forma aplica tanto al lazo de control de corriente como al de velocidad. Además, se adopta la configuración en serie del controlador PI, siguiendo la estructura reportada en las fuentes consultadas para este tipo de sistemas.

Figura 3.6
Lazo de control de corriente



Básicamente, este lazo se controla mediante la cancelación del polo del denominador de la planta mediante el cero del controlador; para esto se igualan los valores de los coeficientes del controlador PI a los de la función que representa la planta. En específico, el coeficiente de la parte integral.

Por lo tanto, para la función de transferencia que relaciona voltaje y corriente en el circuito RL interno del motor.

$$\frac{I(s)}{V(s)} = \frac{\frac{1}{R}}{1+\frac{Ls}{R}} \tag{3.16}$$

Al multiplicar esta relación por el controlador de corriente se obtuvo la función de transferencia en lazo abierto.

$$G(s) = \frac{K_{pi}K_{ii}\left(1+\frac{s}{K_{ii}}\right)}{s}\frac{\frac{1}{R}}{1+\frac{Ls}{R}} \tag{3.17}$$

Donde, si se desea la cancelación de polo y cero, se iguala el coeficiente integral según la ecuación siguiente:

$$Kii = \frac{R}{L} \tag{3.18}$$

Mientras que el coeficiente de la parte proporcional define el ancho de banda luego de hallar la función de transferencia en lazo cerrado, tomando en cuenta la cancelación del polo y del cero correspondientes.

$$G(s) = \frac{1}{\frac{L}{K_{pi}}s+1} \tag{3.19}$$

De la función de transferencia en lazo cerrado se obtuvo el segundo coeficiente del controlador.

$$Kpi = BW_{corriente}L \tag{3.20}$$

Basado en este comportamiento, se podría fijar el ancho de banda ilimitadamente grande, pero esto no es posible debido a limitaciones físicas variadas como pueden ser:

- El sistema no es continuo sino discreto y la discretización depende del tiempo de muestreo.

- Para que el sistema se comporte infinitamente rápido se necesitaría infinita potencia, la cual no está a disposición en la fuente de alimentación.
- El motor solo puede soportar cierta cantidad de corriente, con lo cual también establece límites para su control de corriente.

En el presente caso, un límite crucial es el de discretización, ya que es intrínseco del control mediante microcontroladores y provoca varias no linealidades con las cuales es complicado lidiar con métodos de control en tiempo continuo. Para evitar complicaciones con este límite, se recomiendan comportamientos y respuestas del sistema que sean al menos 10 veces más lentos que el tiempo de muestreo mínimo y, dado que se estimó que se utilizó una frecuencia de muestreo de aproximadamente 10kHz, se utilizó un BW de lazo de corriente de 1 kHz o menores para el presente trabajo.

Basado en lo anterior, se buscaron los coeficientes del controlador de corriente basado en los valores medidos en la sección 4.1 y de las ecuaciones mencionadas anteriormente en esta sección. Para los coeficientes de control, se hallaron los controladores en tiempo continuo y, luego de hallados, se los discretizó para ponerlos en el código del sistema de control.

Se usaron los valores medidos en la sección 4.1, donde los parámetros más importantes fueron:

$$R = 111.3m\Omega \tag{3.21}$$

$$L = 115.6uH \tag{3.22}$$

La calibración se realizó sin tomar en cuenta la discretización y el filtrado que tuvo la variable de corriente por defecto en la librería de código seleccionada, pero se utilizaron criterios para que estos factores no afecten demasiado el comportamiento del control del lazo de corriente. Estos criterios fueron:

- El filtrado de corriente tuvo una frecuencia de corte superior al ancho de banda del sistema y,que será de preferencia, varias veces superior.

- La frecuencia del filtrado de corriente está limitada por la frecuencia de muestreo general del sistema y por el ruido producto de las múltiples señales PWM de voltaje.

Al calcular los valores según las ecuaciones anteriores se obtuvo:

$$Kpi = 0.115 \tag{3.23}$$

$$Kii = 959 \tag{3.24}$$

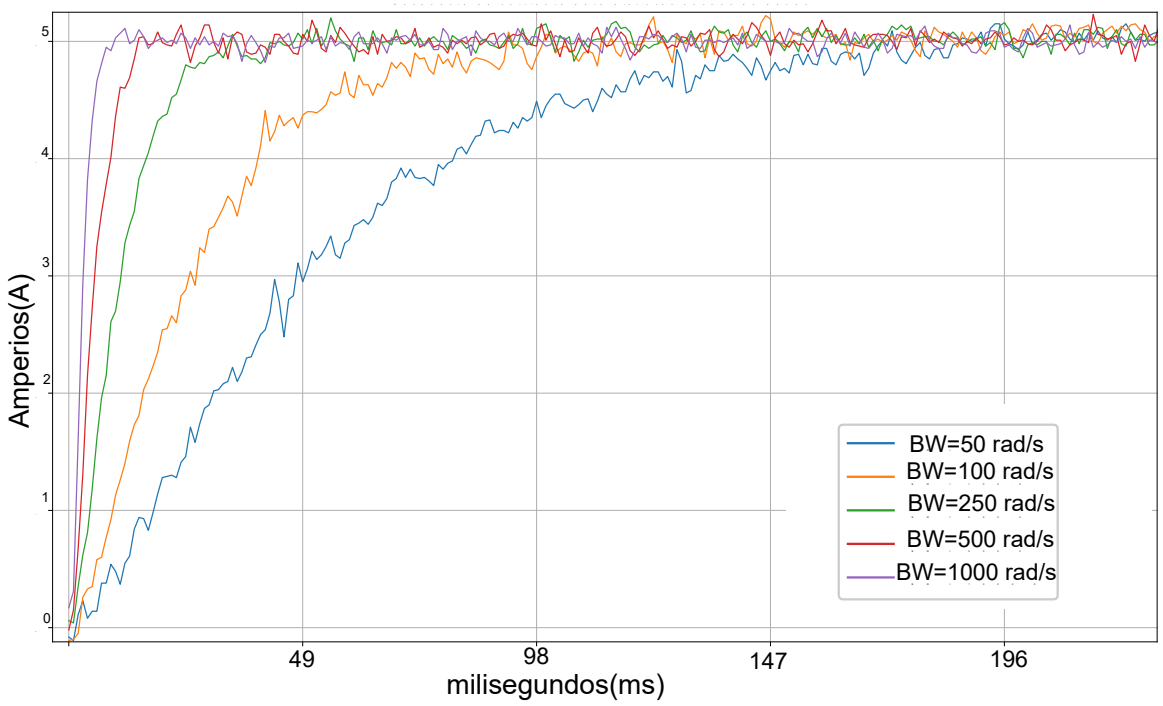
En este punto se realizó un cambio a los coeficientes hallados para poder incorporarlos al código del microcontrolador en valores que tomaran en cuenta el funcionamiento del mismo. Para la adaptación simplemente se pusieron los coeficientes en su forma para controladores PI en paralelo, como lo necesita la librería SimpleFoc (Skuric et al. (2022)). Estos valores se pusieron directamente en el código del controlador y fueron independientes del tiempo, puesto que el tiempo de muestreo ya es calculado internamente y se agrega por defecto en la librería de código. Esto es debido tanto a que el uso de interrupciones complica la configuración de las librerías en los microcontroladores en los que las mismas están adaptadas para funcionar, como a su simpleza y efectividad probada con la prueba y error previos. Por lo tanto, para la conversión de coeficientes se realizaron las siguientes operaciones.

$$P_{corriente} = Kpi = 0.115 \tag{3.25}$$

$$I_{correinte} = Kii \times Kpi = 111 \tag{3.26}$$

Luego de poner los datos calculados para diferentes anchos de banda que van desde 50-1000 rad/s y realizar una prueba de escalón para la corriente del sistema se obtuvieron las gráficas de la figura 3.7. En estas se muestra que los resultados se aproximan a un sistema de primer orden y que el ancho de banda de la curva de estabilización más rápida se mide en 1020 rad/s, lo que cumple con los objetivos del diseño y cálculos anteriores.

Figura 3.7
Diagrama de control de corriente con diferentes anchos de banda



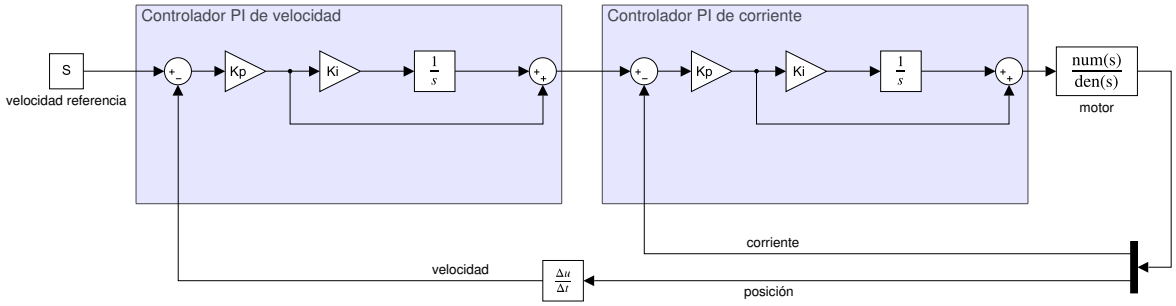
3.9.2. Calibración de lazo de velocidad

Para la calibración previa del lazo de velocidad se utilizó un modelo aproximado y el método visto en Texas Instruments (2021). Esta calibración fue preliminar puesto que no tomó en cuenta:

- Errores de discretización.
- Errores en las mediciones de los parámetros.
- La fricción en los rodamientos del sistema, la cual no se toma en cuenta para el modelo planteado.

Esta calibración inicial sirvió de base para evitar un universo de valores demasiado amplio para los algoritmos de optimización, lo cual habría retrasado la búsqueda o la hubiera hecho demasiado larga al punto de ser totalmente inviable. El diagrama de bloques del sistema aproximado se muestra en la figura 3.8.

Figura 3.8
Diagrama de bloques del lazo de velocidad



La forma del controlador PI es la misma que para el lazo de corriente. Para iniciar con esta calibración se consideró por garantizado el control del lazo de corriente con un ancho de banda de 1 krad/s. Esto sirvió de base ya que no se pueden tener lazos externos con un mayor ancho de banda que los internos y, de no funcionar correctamente el lazo interno, la aproximación como un sistema lineal ideal sería inestable. Estos cálculos buscaron garantizar un punto de estabilidad y fue alrededor del mismo que se realizó la búsqueda de los coeficientes finales.

Debido a la anterior calibración del bloque de corriente, se lo puede simplificar a una simple ganancia donde la función de transferencia entre la corriente y el torque de salida del motor es:

$$\text{Torque} = \frac{3}{4} P \lambda_r \tag{3.27}$$

Donde P es el número de polos del motor y λ_r es el flujo de campo del motor, también igual a la fuerza contraelectromotriz del motor, y el bloque de la carga es representado por:

$$\textbf{carga} = \frac{1}{J}\frac{1}{s} \tag{3.28}$$

Donde J es el momento de inercia del sistema. Combinando todas estas relaciones se obtuvo la función de transferencia de lazo abierto. Además de definir la simplificación K que representa a las constantes del motor.

$$K = \frac{3P\lambda_r}{4J} \tag{3.29}$$

En esta simplificación, al reemplazarla con la variable que relaciona el torque con la corriente K_t que se hallo en secciones previas, se tendrá:

$$K = \frac{Kt}{J} \tag{3.30}$$

Usando esta simplificación para reducir la expresión que representa la función de transferencia en lazo abierto del lazo de velocidad, se obtiene:

$$GH(s) = \frac{K \times Kpv \times Kiv\left(1 + \frac{s}{Kiv}\right)}{s^2\left(1 + \frac{L}{Kpi}s\right)} \tag{3.31}$$

- K es un coeficiente de simplificación antes mencionado.
- Kpv y Kiv son los coeficientes del controlador para el lazo de velocidad.
- L es la inductancia.
- Kpi es el coeficiente proporcional del lazo de corriente.

En este punto se realizó un cálculo de los posibles kp y ki del lazo de velocidad basado en la estabilidad del sistema y el ancho de banda que se podría alcanzar. Para esto, primero se asumió que el siguiente polo del lazo de corriente está a una frecuencia muy superior al cero ($s = Kiv$) del lazo de velocidad.

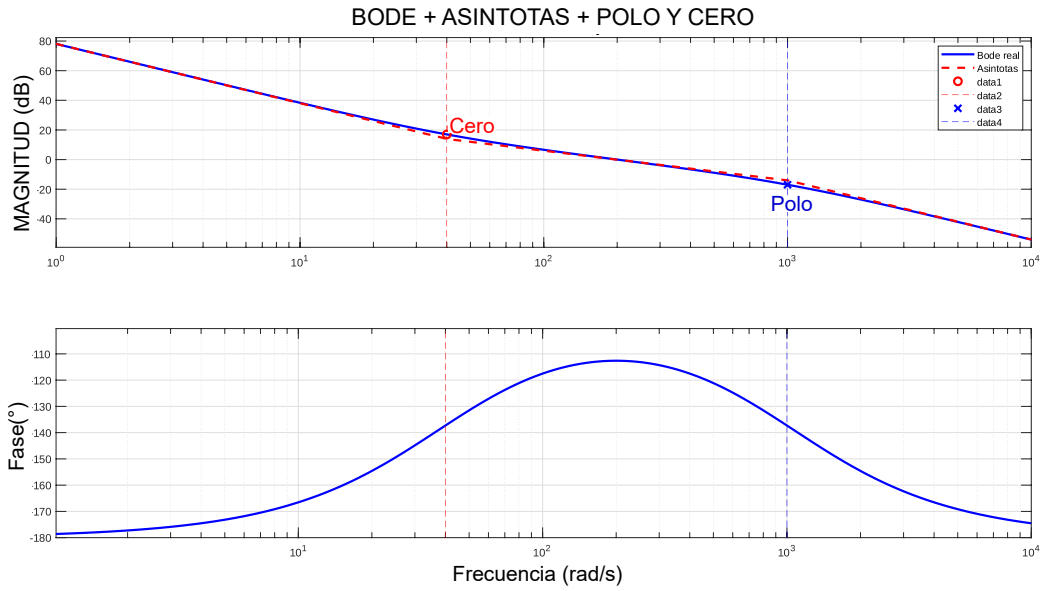
$$s = \frac{Kpi}{L} \tag{3.32}$$

Luego se asumió que el cruce por cero del análisis de bode de la magnitud esta entre estas dos frecuencias, esto es importante ya que este cruce por cero determina la estabilidad del lazo de velocidad.

Para obtener la mayor estabilidad del sistema, este cruce debe ocurrir en la media geométrica de las frecuencias donde se encuentran el cero y el polo, como se muestra en la figura 3.9; basado en esto, se establecen las relaciones.

Figura 3.9

Bode de ubicación del cruce por cero deseado



$$\omega_{\text{polo}} = \delta \times \omega_0 \tag{3.33}$$

$$\omega_{\text{polo}} = \delta^2 \times \omega_{zero} \tag{3.34}$$

Donde se introduce un nuevo parámetro δ, el cual se define como el factor de amortiguamiento que relaciona la estabilidad y el ancho de banda. Idealmente, a mayor factor de amortiguamiento, mayor estabilidad y menor ancho de banda, lo mismo en viceversa. Adicionalmente; de manera teórica, cualquier valor del factor mayor a 1 deberá ser estable.

Dado que W_0 ya está definido en términos de Kpi y Kii del lazo de corriente, con una sustitución del valor w0 en las ecuaciones anteriores se obtiene:

$$Kiv = \frac{Kpi}{\delta^2 L} \tag{3.35}$$

Luego, para hallar el último valor faltante, se aprovecha el cruce por cero, punto donde la

magnitud de la función de transferencia de lazo abierto tiene que ser 0 dB o uno en términos absolutos. Se obtiene de esta forma la siguiente ecuación.

$$\left| \frac{K \times Kpv \times Kiv \left(1 + \frac{s}{Kiv}\right)}{s^2 \left(1 + \frac{s}{\delta^2 \times Kiv}\right)} \right|_{s=j\delta \times Kiv} = 1 \tag{3.36}$$

Tras realizar el reemplazo de la variable s, se obtuvo:

$$\frac{\delta \times K \times Kpv}{\delta^2 \left(\frac{K_{pi}}{\delta^2 \times L}\right)} = 1 \tag{3.37}$$

Donde finalmente se pudo despejar el valor de Kpv:

$$Kpv = \frac{Kpi}{LK\delta} \tag{3.38}$$

Reemplazando el valor de Kpi por su equivalente en términos de Kiv, se obtuvo otra expresión para el mismo valor.

$$Kpv = \frac{\delta \times Kiv}{K} \tag{3.39}$$

Con estos cálculos se obtuvieron los valores iniciales proporcionales e integrales de los controladores de velocidad; si bien estos cálculos son totalmente ideales puesto que no toman en cuenta la fricción de los componentes o los comportamientos no lineales del motor y sistema (límites físicos de velocidad, aceleración, voltaje o corriente del sistema), se usaron como una aproximación inicial sobre la cual realizar la optimización, tal como se especificó en las reglas planteadas a partir de estas bases en la sección de algoritmos de optimización.

En este punto se observa que el valor de δ o factor de amortiguamiento es arbitrario siendo el único requisito teórico que sea mayor que 1; sin embargo, para limitarlo y darle un valor base a partir del cual se realicen las optimizaciones, se tomaron los criterios que

relacionan a las variables del lazo de velocidad y corriente de Texas Instruments (2021), los cuales proporcionan un modo de relacionar los componentes de ancho de banda de corriente y de velocidad mediante una ecuación . Además, se usó el criterio de lazos acotados, según el cual un lazo externo debe tener un ancho de banda de 5 a 10 veces más lento que uno interno, como se muestra en Landolfi and Natale (2023), Wang et al. (2022) y Microchip Technology Inc. (2023). Por lo tanto, se plantea la siguiente ecuación.

$$\mathrm{BW}_{corriente} = \frac{K_p^{\text{series}}}{L} = \mathrm{BW}_{velocidad}\left(\delta + 2.16\,e^{-\frac{\delta}{2.8}} - 1.86\right) \tag{3.40}$$

Y los valores iniciales.

- BW de corriente: 1000 rad/s.
- BW de velocidad: 100 rad/s.

Se obtuvo un valor de:

$$\delta = 11.83 \tag{3.41}$$

Este valor está dentro de lo descrito anteriormente, puesto que se necesita que δ sea mayor que 1 para que el sistema sea idealmente estable; por lo tanto, bajo criterios ideales, el sistema es estable. Con este valor delta, los anchos de banda designados, la inductancia y resistencia del motor, se obtienen los coeficientes del controlador de velocidad.

$$Kiv = \frac{Kpi}{\delta^2 L} = 7.15 \tag{3.42}$$

$$Kpv = \frac{\delta \times Kiv}{K} = 28.9 \tag{3.43}$$

Por último, se convirtieron estos coeficientes en su forma en paralelo, como se realizó

con los coeficientes para el controlador de corriente, para que fuesen compatibles con el código de la librería de control.

$$P_{velocidad} = Kpv = 28.9 \tag{3.44}$$

$$I_{velocidad} = kiv \times kpv = 206 \tag{3.45}$$

Si bien con estos valores se puede garantizar la estabilidad del sistema y estos pueden servir para pruebas previas del lazo de posición para hallar un rango de valores donde realizar la búsqueda, se realizó un cálculo extra para limitar los valores del campo de búsqueda ; para esto se mantuvieron los valores de los anchos de banda en:

- BW de corriente: 1000 rad/s.
- BW de velocidad: 100 rad/s.

A continuación, en lugar de hallar solo el valor del factor de amortiguamiento para lograr las condiciones de ancho de banda, se buscó el rango de valores donde el factor de amortiguamiento fuese mayor que 1.5, es decir, de este valor hasta un valor tendiente a infinito. Se utilizó este rango, debido a que, según el modelo de la ecuación (3.40), se garantiza la estabilidad para una planta lineal e idealmente mantiene el sobreimpulso en valores reducidos. De estas operaciones se obtiene los rangos de búsqueda siguientes para los coeficientes del controlador.

$$P_{velocidad} = 34.1 - 20.6 \tag{3.46}$$

$$I_{velocidad} = 0.01 - 827 \tag{3.47}$$

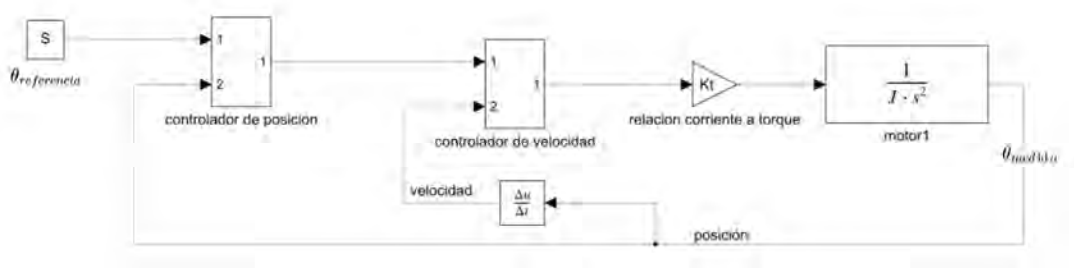
Se utilizaron estos rangos multiplicados por un factor de 1.5 como medida adicional arbitraria para la búsqueda de los coeficientes que mejor reduzcan la función objetivo.

3.9.3. Calibración de lazo de posición

La calibración del lazo de posición se realizó con ayuda del modelo matemático, el cual tiene los valores antes calculados para los lazos internos y que muestra cómo afecta el comportamiento del coeficiente del lazo externo de posición en el comportamiento general. Esta parte es un análisis heurístico basado en los modelos matemáticos que ayudan a poner límites y características al coeficiente P del lazo más externo de nuestro controlador.

Para el lazo de posición completo, se tomó como base la aproximación de los lazos interiores vista en el trabajo de Mandra (2014). Estas aproximaciones de los bloques de lazo se muestran en la figura 3.10.

Figura 3.10
Diagrama de bloques completo

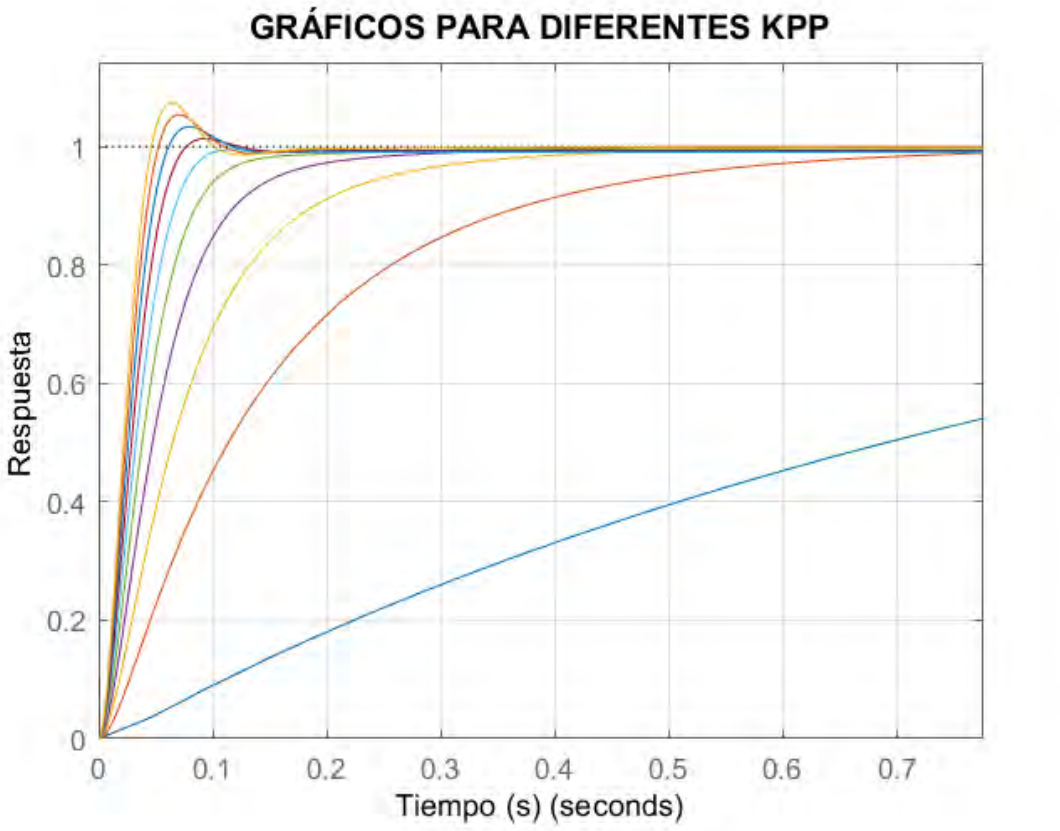


En esta aproximación se observa que se toma el lazo de corriente como una relación lineal de la corriente de entrada con el torque producido por el sistema, de este diagrama de bloques se obtiene la función de transferencia de lazo cerrado que relaciona el ángulo objetivo del sistema con el ángulo medido en el mismo:

$$G(s) = \frac{\theta_{medido}(s)}{\theta_{referencia}(s)} = \frac{(K_{pp}K_{pv})s + K_{pp}K_{iv}}{\frac{J}{K_t}s^3 + (K_{pv})s^2 + (K_{iv} + K_{pp}K_{pv})s + K_{pp}K_{iv}} \tag{3.48}$$

Basado en la medición aproximada de valores realizada en las secciones anteriores y en las ecuaciones planteadas para el lazo de velocidad, se reemplazaron los valores, obteniendo así una expresión que solo depende del factor proporcional del lazo de posición. Probando su respuesta al escalón con diferentes valores de la constante proporcional que van desde 0.1 a 2, se obtuvo el gráfico de la figura 3.11.

Figura 3.11
Respuestas para diferentes P de posición



Este gráfico muestra que el comportamiento del lazo de posición es idealmente estable en cada ocasión, con la única diferencia de que, a mayor valor de la constante proporcional, menor tiempo de establecimiento y para valores muy altos existe la aparición de sobreimpulso. Esto indica que se obtiene el mejor comportamiento posible si se agranda este factor hasta cierto punto. Además, todo tiene que funcionar dentro de un sistema real que tiene límites físicos ligados al límite de corriente que se puede aplicar para producir torque. La simulación utiliza una aproximación lineal sin límites para la magnitud de la señal de control y no se toma en cuenta otros factores como la corriente máxima que puede soportar el motor; por lo tanto, incrementar la constante proporcional no siempre es sinónimo de mejora en el comportamiento.

Para definir un límite al espacio de búsqueda, a diferencia del caso del lazo de velocidad, se empleó una prueba heurística donde se utilizaron los valores base hallados en la sección anterior para el lazo de velocidad.

$$P_{velocidad} = Kpv = 28.9 \tag{3.49}$$

$$I_{velocidad} = kiv \times kpv = 206 \tag{3.50}$$

Con estos valores que garantizan estabilidad, se realizó la siguiente prueba para el coeficiente de control de posición.

- Se puso el valor de la variable en 1 y se envió una prueba escalón del valor de optimización que se utilizará, en el caso de la presente investigación es 3.14.
- Se observó el comportamiento del sistema; de no tener sobreimpulso en su movimiento, se incrementaba en una unidad el valor del lazo y se repetía la prueba escalón.
- Cuando se observó un sobreimpulso visible, se detuvo la prueba y se tomó este valor como base de la optimización.

Como en el caso del lazo de velocidad, se utilizó este valor multiplicado por 1.5 como el límite del rango donde optimizar el coeficiente del lazo de posición.

3.10. Selección de motor, placa de control y potencia:

En esta parte del trabajo se seleccionaron los componentes físicos que componen la planta de control; además, se muestran los límites físicos y características que tienen los componentes seleccionados.

El sistema general consta del motor, la placa de control y el encoder. La placa de control incluye un microprocesador que funciona como la unidad lógica y una unidad de potencia que está conectada al motor directamente.

3.10.1. Selección del motor para el armado de la planta

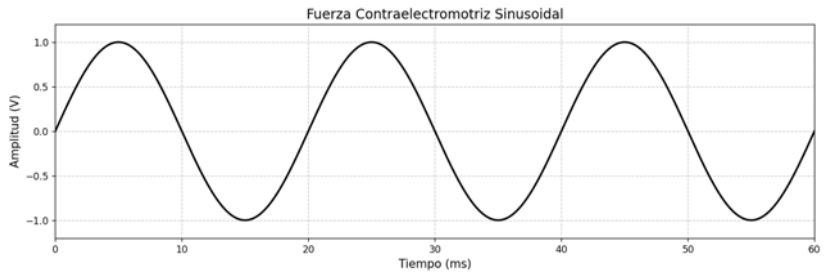
El control de posición se realizó con un motor sin escobillas de onda contraelectromotriz sinusoidal (Mohanraj et al. (2022)). También se le puede llamar motor PMSM según la fuente que se consulte. En el presente trabajo se lo menciona con el primer nombre.

Las características que debe cumplir el motor utilizado son:

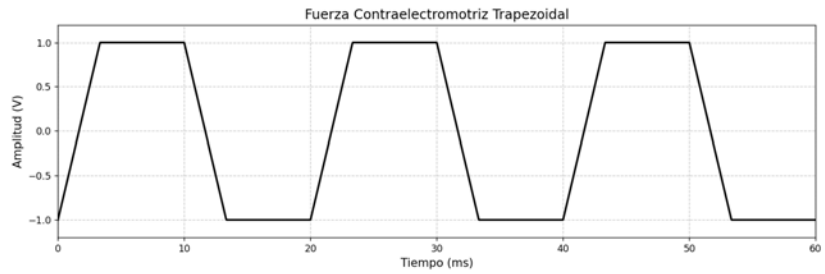
- Tener una onda contraelectromotriz de forma sinusoidal.
- Resistir un voltaje de entrada de 12V, 24V o 48V, que son los voltajes comunes usados en fuentes DC.

El resto de características de este tipo de motores, como no poseer escobillas, tener una entrada de tres fases y devanado interno conectado en forma de estrella, las cumplen todos los motores estándar y por eso no se las menciona. Los dos tipos de onda contraelectromotriz se muestran en la figura 3.12.

Figura 3.12
Onda contraelectromotriz sinusoidal (a) y trapezoidal (b)



(a)



(b)

Las opciones para escoger fueron aquellos motores que especificaban que están hechos para este tipo de aplicación o que pertenecían a la clasificación de motores para Vehículo

Aéreo No Tripulado (VANT) agrícola, esto porque los mismos cumplen con las características básicas antes mencionadas y pueden proveer un torque considerable, ya que no están diseñados para girar a altas velocidades, sino para llevar alta carga con hélices muy grandes que giran a bajas revoluciones para un dron promedio. Como ejemplos de estos motores en aplicaciones relacionadas tenemos el trabajo Ruppert and Badri-Spröwitz (2022), donde se usaron este tipo de motores para accionar los actuadores de un robot cuadrúpedo. Ejemplos de este tipo de motores se aprecian en la figura 3.13.

Figura 3.13
Posibles motores para la investigación

(a) *Motor diseñado para actuadores*

Nota. *Adaptado de Eaglepower (nd)*

(b) *Motor diseñado para VANT agrícola*

Nota. *Adaptado de Cube Mars (nd)*

(c) *Motor diseñado para VANT agrícola*

Nota. *Adaptado de Robotics (nd)*

El motor escogido en específico fue el Eaglepower LA 8308 KV90 por ser parte de los motores de VANT tipo agrícola antes mencionados, tener un precio bajo en comparación a los otros, además de estar comercialmente disponible en tiendas virtuales. Este motor está pensado para ser usado en VANT agrícolas, pero estas características también le dan lugar en aplicaciones que lo utilizan como un actuador. Las características básicas de referencia de este motor se encuentran en la figura 3.2.

Tabla 3.2
Características del motor

Comparible LIPO Batteries	6S TO 12S
Idle Current(A)	0.6A at 24V
Size	92*28.5mm
Weight(g)	336
Continuous Current(A)	22
Max Power(W)	900
Internal Resistance(Ohm)	0.186

3.10.2. Placas de control lógico y de potencia

Las placas de control de potencia fueron parte fundamental de esta investigación, ya que de su funcionamiento y configuración dependió el éxito de las pruebas de los algoritmos de optimización. Para su selección se tomarán los siguientes criterios a consideración.

- Amperaje y voltaje que son capaces de manejar.
- Microcontroladores con los que son compatibles.
- Sensores que tienen incluidos o con los que son compatibles.
- Código abierto o librerías para su configuración.
- Finalmente, factor económico y disponibilidad.

Además, las placas deberán cumplir los siguientes requisitos mínimos para el desarrollo del presente trabajo.

- Poseer tres medios puente H para el control independiente de cada fase.
- Aceptar una alimentación de 12 V, 24 V o 48 V, siendo estos los voltajes estándar de las fuentes DC.
- Tener integrados al menos dos medidores de corriente de fase, con los cuales se pueda determinar el valor de la corriente de la tercera fase al estar esta relacionada con los anteriores.

Se seleccionó entre las siguientes placas mostradas en las figuras 3.14a, 3.14b, 3.14c y 3.14d, que se tienen disponibles en el mercado y cumplen con las características mínimas requeridas.

Figura 3.14
Posibles placas para la investigación

(a) *SimpleFOCShield v3.2*

Nota. *Adaptado de SimpleFOC Project (2024)*

(b) *MKS ESP32 FOC V1.0*

Nota. *Adaptado de Makerbase (2024)*

(c) *MKS XRIVE MINI de Makerbase*

Nota. *Adaptado de Makerbase (2023)*

(d) *STEVAL-SPIN3201*

Nota. *Adaptado de STMicroelectronics (2021)*

De las placas listadas se escogió la MKS XRIVE MINI de Makerbase por los siguientes motivos.

- Soporta más corriente y potencia que las otras. Si bien, cuando trabaja a máxima capacidad, necesita ventilación directa, esto no ocurre con las pruebas de experimentación del presente trabajo, ya que la placa no conduce más de 20 A de corriente, con la cual no eleva su temperatura de manera considerable.
- Si bien no es compatible con muchos microcontroladores, la familia de los STM32 es de los controladores comunes más potentes del mercado al tener procesadores de

Tabla 3.3
Comparativa de placas para control FOC

Placa	Tensión y corriente	MCU	Sensores compatibles	Código y librerías	Precio y disponibilidad
SimpleFOC Shield v3.2	8-35V, 3A máximo (con disipador y ventilación)	Atmega (Arduino), STM32 (NUCLEO)	Encoder magnético (AS5600, AS5047), Encoder de cuadratura 5V	Librería SimpleFOC (código abierto)	S/.63.09 Disponible en ALIEXPRESS
Makerbase ESP32 Dual SimpleFOC	12-24V, 12A máximo (con disipador y ventilación)	ESP32 WROOM	Encoder magnético (AS5600)	Librería SimpleFOC (código abierto)	S/.81.72 Disponible en ALIEXPRESS
MKS XRIVE MINI Makerbase	12-56V, 60A RMS (con disipador y ventilación)	STM32F405TGT6	Encoder magnético (AS5047 incluido), Encoder cuadratura 5V	Librería SimpleFOC (código abierto), Librería ODRIVE (propietario)	S/.159.34 Disponible en ALIEXPRESS
STEVAL-SPIN3201	8-45V, 15A RMS	STSPIN32F0	Encoder de cuadratura	Motor Control SDK (X-CUBE-MCSDK, propietario STMicroelectronics)	S/.63.96 Disponible en STMicroelectronics

32 bits; además de tener ADCs de 12 bits, los cuales tienen UNA mejor precisión comparándolos con los del ESP32.

- Tiene un encoder magnético incluido de tipo AS5047, el cual se comunica mediante el protocolo SPI con una frecuencia de reloj de hasta 1 MHz.
- Tiene compatibilidad con librerías de código abierto como SimpleFOC y de código propietario como ODRIVE.
- La placa tiene un costo intermedio comparado con las demás opciones y se ahorra la necesidad de un encoder, el cual necesitarían las demás placas de manera obligatoria.

3.11. Descripción de la placa y componentes internos

En esta sección se procedió con la descripción de la placa MKS XRIVE MINI y los componentes internos que se necesitaron para realizar el control vectorial de un motor sin escobillas.

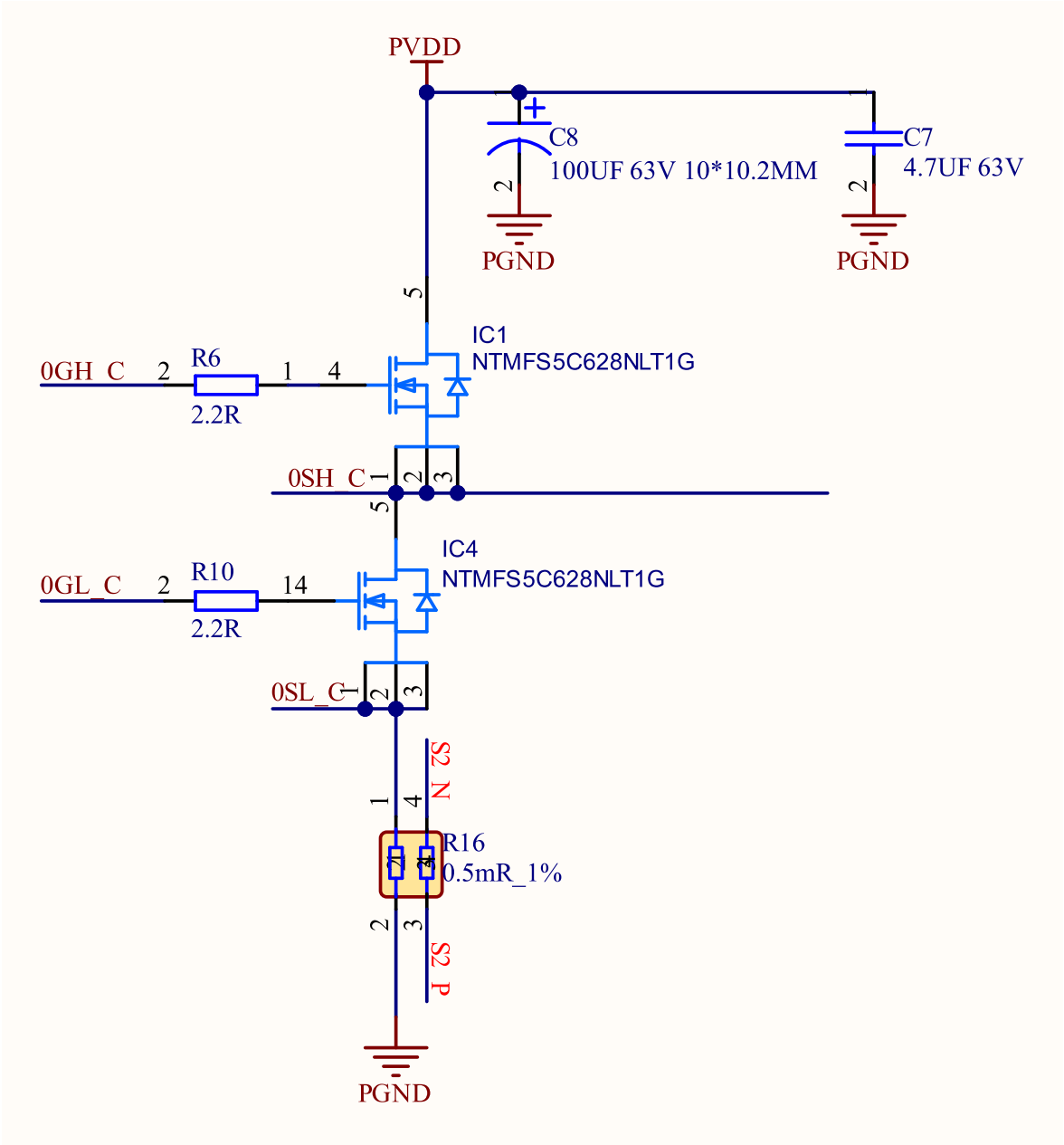
3.11.1. Medios puentes H

En la placa seleccionada, los transistores que componen el control de potencia son del tipo NTMFS5C628NLT1G que, como principales características, tienen:

- Tolerancia a una corriente de drenador pico de hasta 150 A a 25 grados centígrados.
- Voltaje de drenador a fuente de hasta 60 V máximo.

Estas características cumplen con lo ofrecido por la empresa que diseñó la placa y, por lo tanto, no se tuvieron problemas con el apartado de potencia de la placa de control. La configuración de los transistores para cada medio puente se aprecia en la figura 3.15. Donde se aplicó la misma estructura de interruptores ideales presentada teóricamente en la figura 2.11. Por otro lado, Una resistencia en serie con la compuerta limita el pico de corriente que entrega el controlador, reduce las oscilaciones y el ruido eléctrico; los condensadores cumplieron principalmente la función de estabilizar el bus de alimentación de corriente continua. Actuaron como un depósito de energía que suministró las corrientes instantáneas demandadas por los MOSFET durante la conmutación y absorbieron los picos de corriente generados por las cargas inductivas, lo que redujo el rizado de la tensión y evitó caídas bruscas en el voltaje de alimentación.

Figura 3.15
Medio puente H de la placa de control



Nota. *Adaptado de Makerbase (2023)*

3.11.2. Resistencias Shunt de 0.5 mOhm

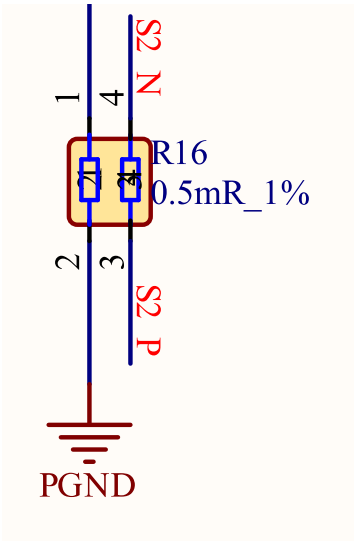
Como se puede observar en la gráfica 3.16 se tienen resistencias shunt en la parte baja de la configuración del medio puente H, esta resistencia sirve para medir la corriente que circula por las fases y se necesitan, por lo menos, dos de estas resistencias en los 3 puentes H para saber los valores de todas las corrientes, ya que se encuentran relacionadas por la ecuación

de corrientes en un sistema trifásico.

$$IA + IB + IC = 0 \tag{3.51}$$

Como se muestra en la ecuación anterior. Se puede obtener una de ellas despejando los valores de las otras dos.

Figura 3.16
Resistencia shunt para medición de corriente



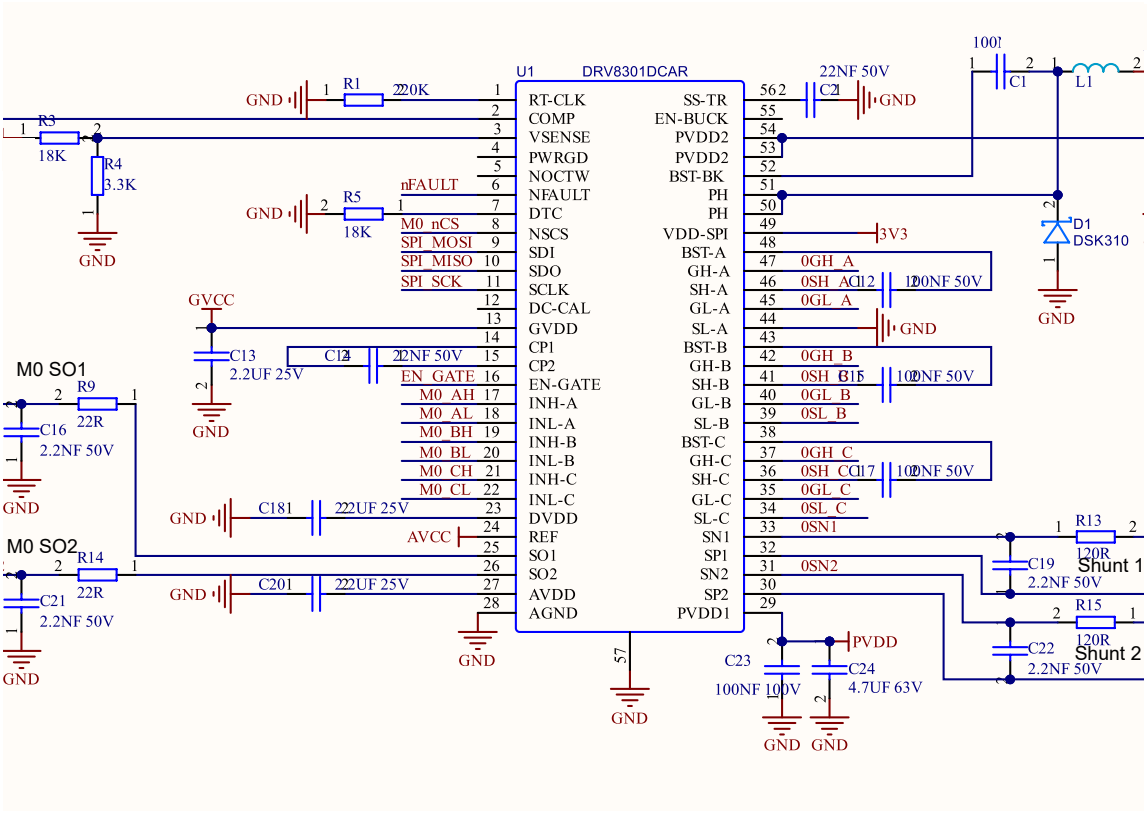
Nota. *Adaptado de Makerbase (2023)*

3.11.3. Driver de transistores DRV8301DCAR

Este driver es un componente extensamente usado para el control de motores sin escobillas, este se ocupa tanto de la gestión de las conmutaciones de los 6 transistores de potencia como de la medición de la corriente, teniendo amplificadores operacionales internos a los cuales se les conecta el voltaje entre las resistencias shunt, como se puede apreciar en la gráfica 3.17, de manera específica, en la parte inferior derecha se tienen dos entradas con filtros paso bajo para las señales de voltaje provenientes de las resistencias shunt.

En su configuración, este driver además integra dos filtros paso bajo en la salida del driver hacia el microcontrolador, ya que la señal con la que trabaja es analógica. La frecuencia de corte de ambos filtros es:

Figura 3.17
Driver de los 3 medios puentes H



Nota. *Adaptado de Makerbase (2023)*

$$\frac{1}{2\pi \times 120(2.2nf)} = 602KHz \tag{3.52}$$

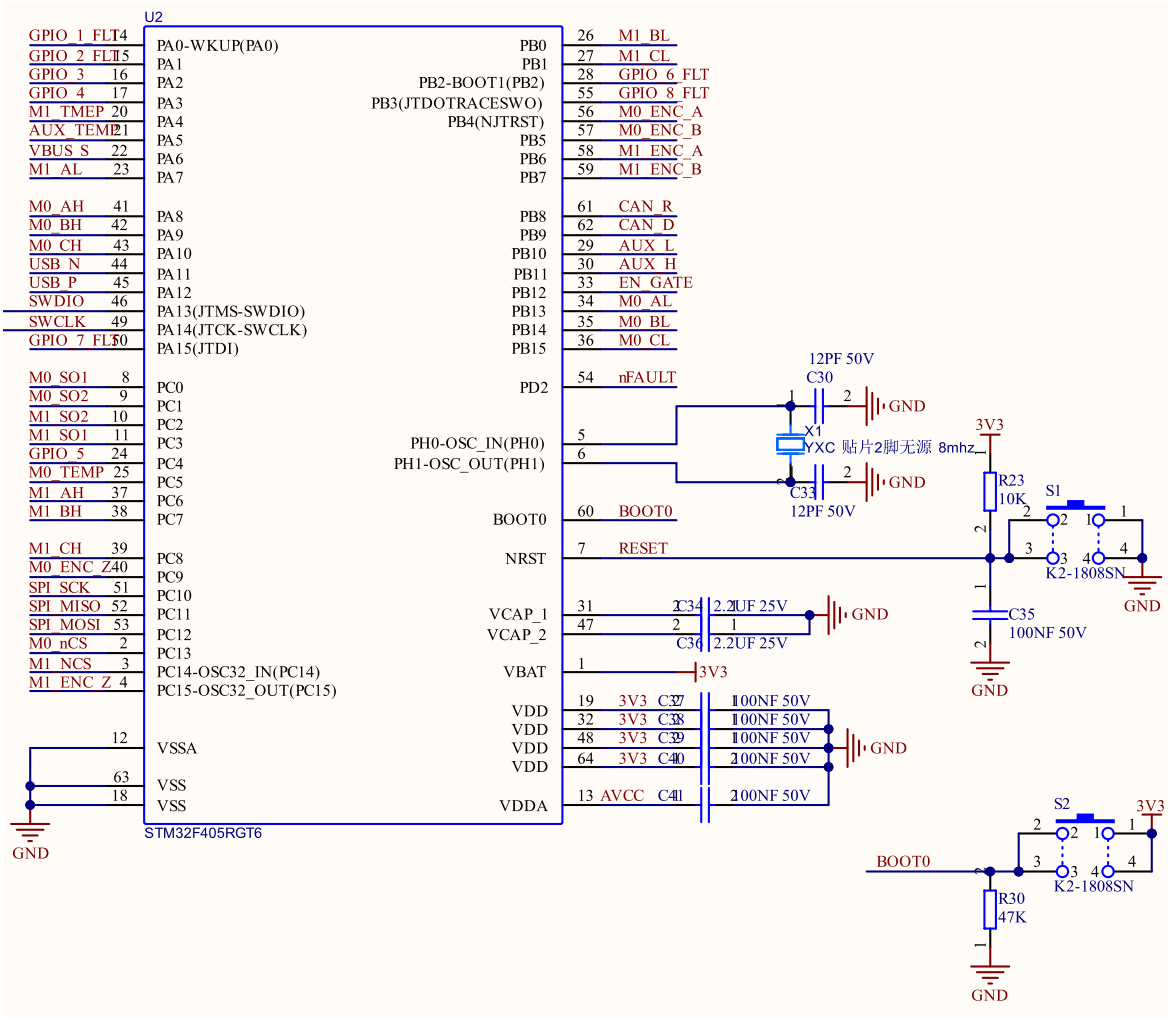
$$\frac{1}{2\pi \times 22(2.2nf)} = 3.228MHz \tag{3.53}$$

Estas dos frecuencias disminuyen el ruido de alta frecuencia que se puede producir en la lectura de corriente, pero al tener una frecuencia de corte tan alta, no afectan las lecturas del microcontrolador. Además, las señales de corriente deberán filtrarse nuevamente en la etapa lógica mediante filtros digitales, ajustados según el ancho de banda de corriente requerido y el nivel de ruido observado en las mediciones, con el fin de obtener una estimación más estable y precisa.

3.11.4. Microcontrolador STM32F405

Se presentó el microcontrolador incluido en la placa con sus periféricos en una configuración básica para su funcionamiento. Este incluye un cristal de 8 MHz y las configuraciones básicas para su programación, como los botones de Boot y Reset, tal como se muestra en la figura 3.18. Asimismo, la placa incorpora otros periféricos que complementan su funcionamiento, como la interfaz de comunicación serial a USB para la programación, depuración y monitoreo del sistema, la cual no se muestra directamente en la figura. Estos elementos permiten establecer la comunicación entre el microcontrolador y el computador durante las etapas de desarrollo y pruebas, facilitando la carga del firmware y la visualización de datos.

Figura 3.18
Microcontrolador



Nota. *Adaptado de Makerbase (2023)*

3.11.5. Encoder magnético AS5047

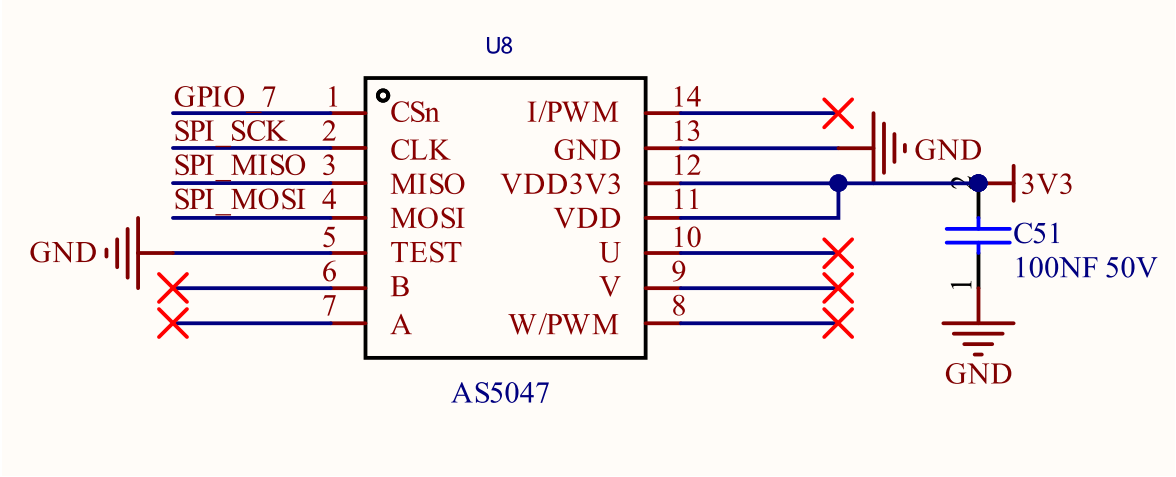
Este es un encoder de tipo magnético, por lo que requiere un imán de polarización radial pegado al eje de rotación del motor y, mediante la medición de los campos generados por este imán, se determina la posición del rotor.

Este encoder tiene las siguientes características principales.

- Resolución de 14 bits.
- Tipo de encoder absoluto.
- Voltaje de funcionamiento 3.3V.
- 1 MHz de frecuencia máxima de pulso de reloj.
- Tipo de comunicación SPI.

El tipo de comunicación y conexiones básicas de este encoder se aprecian en la figura 3.19.

Figura 3.19
Encoder



Nota. *Adaptado de Makerbase (2023)*

3.11.6. Diseño en 3D para acoplamiento de componentes

Dados los componentes seleccionados, se necesitaron bases o soportes que sean compatibles y de precisión. Esto es importante en el acoplamiento del encoder en el eje del motor, puesto que errores en este acoplamiento pueden causar errores de medición de ángulo críticos, esto se debe a que el encoder utilizado tiene que ser colocado en el centro del eje de rotación del motor a una distancia de unos pocos milímetros. Para solventar estos errores y, además, tener una base que acople los componentes de la planta, se usó la impresión en 3D con diseños específicos para estos propósitos mostrados en la figura 3.20. Se diseñaron piezas que acoplen los siguientes componentes:

- Motor y encoder incluido en la placa de control y de potencia.
- Rotor de motor y barras de metal que sostienen las cargas.
- Unión de las barras de metal a las pesas que simulan la carga del sistema.
- Soporte de imán pegado al rotor del motor.

El diseño se realizó en el software Fusion360 y se imprimieron los componentes en una impresora Creality Ender 3 V3 SE. Este diseño tuvo el propósito de sujetar los componentes y, dado que no se expusieron las piezas a grandes presiones o deformaciones, no se tomaron en cuenta factores como el nivel de deformación o carga que soportan, asumiendo estos valores como insignificantes durante el diseño.

Adicionalmente, se consideró que la tolerancia dimensional de la impresión 3D podría introducir pequeñas variaciones en el ajuste final de las piezas, por lo que se dejaron márgenes de holgura controlados en las uniones mecánicas. Esto permitió asegurar un ensamblaje adecuado sin comprometer la alineación entre los distintos elementos del sistema. Asimismo, el uso de este tipo de fabricación facilitó la iteración rápida del diseño en caso de ser necesario realizar ajustes posteriores. Finalmente, se priorizó la simplicidad estructural para reducir posibles puntos de falla y mejorar la estabilidad general del montaje.

El sistema completo y ensamblado de uniones 3D se ve como en la figura 3.21.

Figura 3.20

Modelos 3D de piezas diseñadas para el soporte del motor BLDC y sus componentes

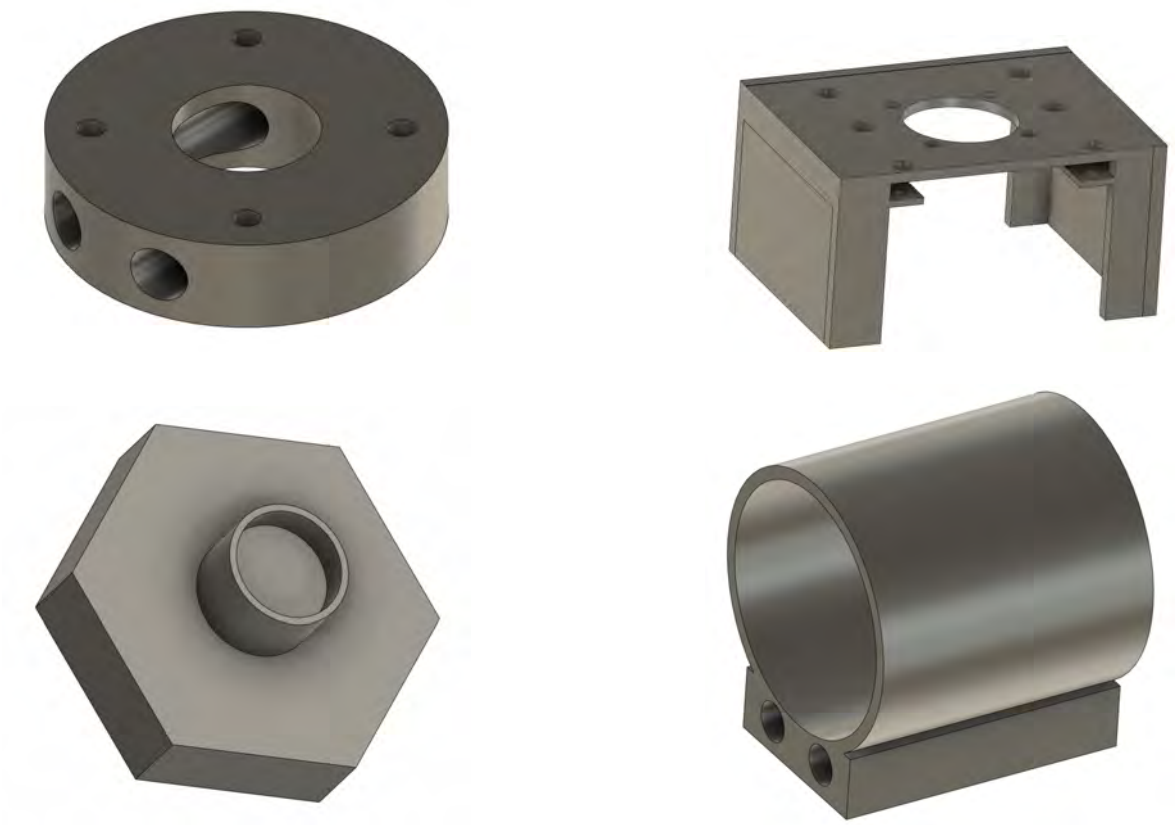
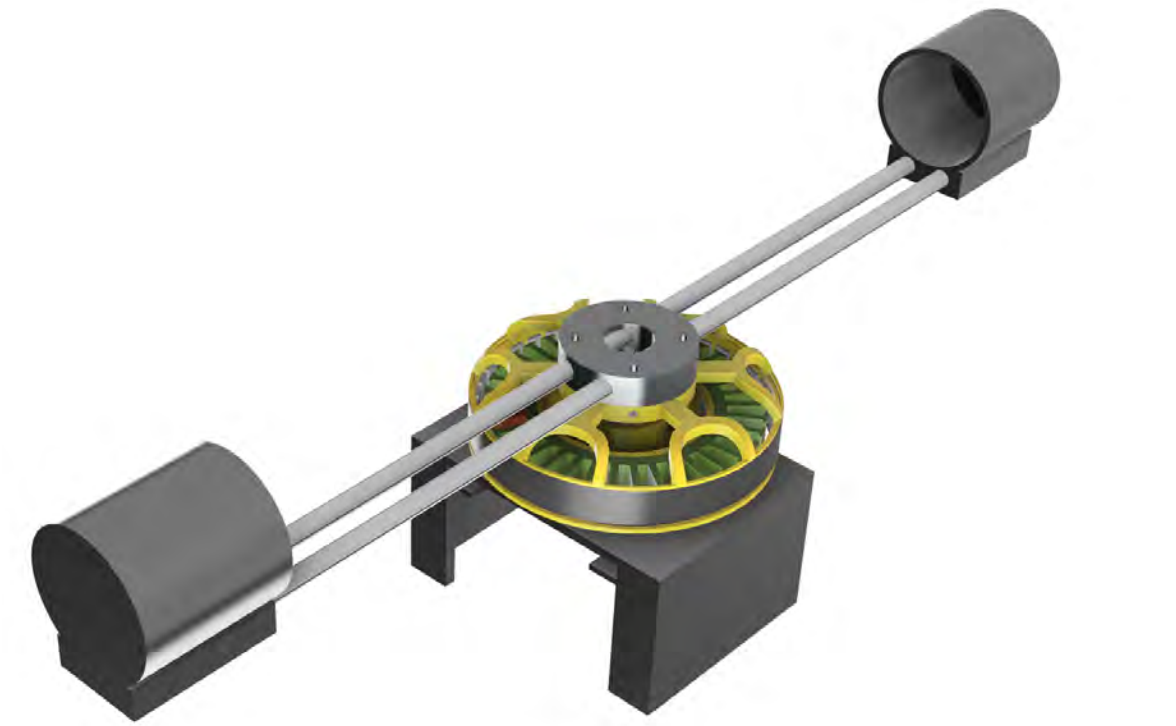


Figura 3.21

Modelo 3D de soportes ensamblados



3.12. Calibración de sensores de corriente y encoder de la placa de control.

Las dos variables fundamentales que el sistema necesita para el control son la corriente por las fases y el ángulo de posición del rotor; por lo tanto, el sistema tiene sensores para estas dos variables que son las resistencias shunt con amplificadores operacionales y el encoder magnético AS5047. Para corroborar la correcta medición de estas variables, se realizaron calibraciones.

3.12.1. Calibración de censado de corriente

Para esta calibración se comparó la medida de la corriente captada por el sistema con la medida de un multímetro patrón que es el modelo UT210E de la marca UNI-T, que posee un amperímetro con pinza y tiene las siguientes características:

- Rango de medición de 0 a 100 A.
- Resolución de 5 mA.
- Mínima unidad de medición de 1 mA.

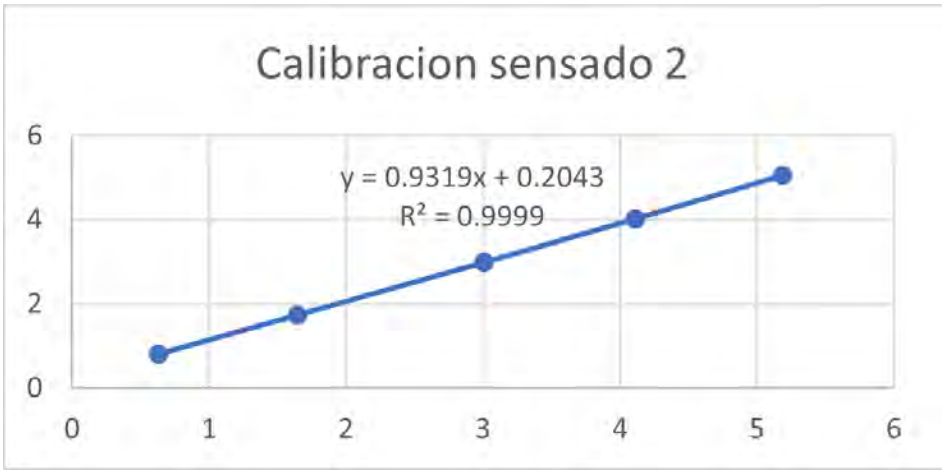
Para esta medida se realizó el siguiente proceso:

- Se enviaron pulsos de voltaje al sistema hasta llegar a un valor cercano a 5 A medidos por el multímetro.
- Se obtuvo, mediante el puerto serial, la medida del sistema de la corriente generada por estos pulsos; al mismo tiempo, se puso a medir la corriente que pasa por los cables de alimentación mediante el multímetro.
- Se tomaron 70 valores medidos por la placa y se los promedió para evitar el ruido de corriente del sistema.

- Se compararon los resultados obtenidos para los dos sensores internos de la placa y los valores reales medidos por el multímetro, obteniéndose los gráficos 3.22 y 3.23.

Figura 3.22
Curva de calibración para el sensor 1

Figura 3.23
Curva de calibración para el sensor 2



En los gráficos se muestra una relación lineal de las medidas realizadas mediante los sensores internos y la del multímetro. Teniendo en cuenta estos valores, se corrobora que el sistema sí cumple con el sensado de la variable de corriente; además, se pueden realizar pequeños ajustes de las ganancias para evitar los errores de medición.

3.13. Calibración de encoder

La calibración del encoder se realizó para la comprobación del correcto funcionamiento del sensor puesto que el mismo podía tener averías o estar desalineado del eje de rotación

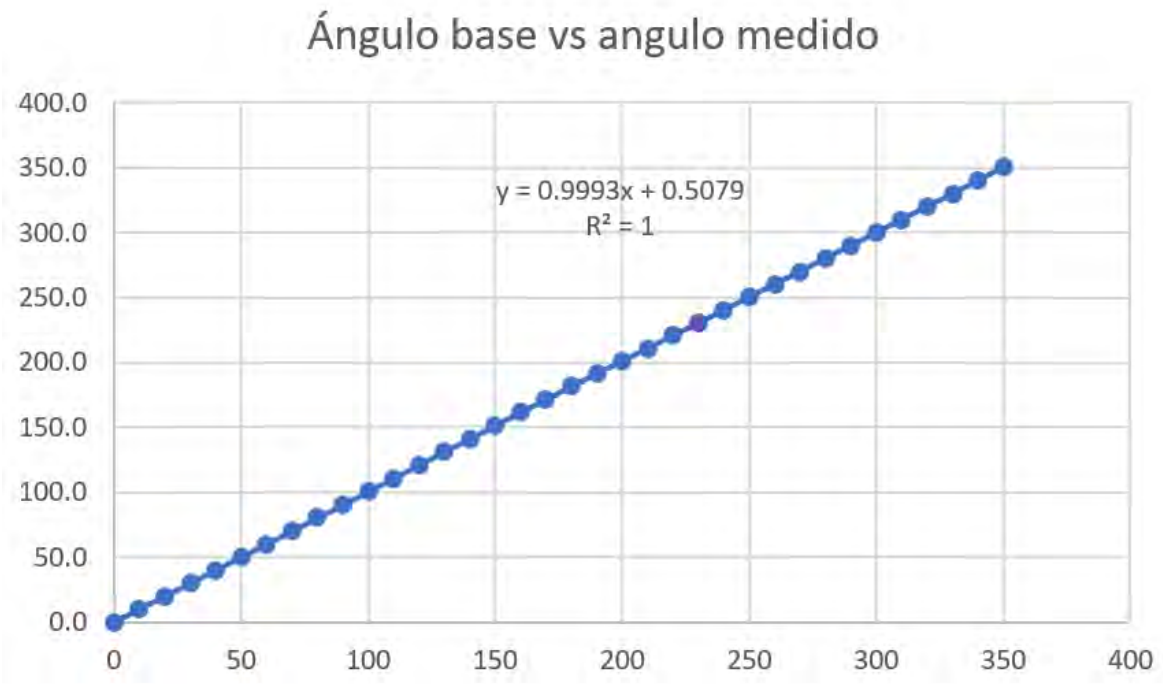
del sistema causando errores considerables.

Para realizar la calibración se ejecutó el siguiente procedimiento:

- Se une al eje central del motor un transportador de ángulos.
- Se realizan medidas cada 10 grados, desde los 0-350 grados.
- Se contrastan las medidas visuales con las medidas que mostró en cada punto el encoder absoluto magnético.

La tabla de resultados se encuentra en los anexos del presente trabajo y el gráfico 3.24 muestra la relación de las medidas patrón y las obtenidas del sensor.

Figura 3.24
Comparación y relación de medidas de ángulos base y de encoder



Basado en el gráfico y las medidas, se observó un error máximo de 1.6 grados sexagesimales el cual no es recurrente siendo el error promedio real 0.4 grados, estas medidas de error al representar menos del 1 % de error se tomaron por ideales.

3.14. Códigos para el manejo de la placa de control y potencia

Para la programación y desarrollo de los algoritmos se utilizó una librería de código abierto de nombre SimpleFOC (Skuric et al. (2022)). Esta librería se empleó en modo de control de la variable de posición para un motor sin escobillas. Esta librería es compatible con la placa seleccionada y mediante su uso se ahorra la escritura de código puro para realizar el control y permite el enfoque del trabajo en los coeficientes de los controladores, ademas se tiene la libertad de analizar las funciones que la componen al ser de código abierto. Esta librería se encarga, en específico, de los procesos de:

- Lectura de señales analógicas de corriente.
- Lectura del encoder.
- Estructuración de los lazos de control, desde el de corriente al de posición.
- Acoplamiento de filtros de paso bajo digitales para cada lazo de control.
- Transformaciones Clark y Park directa e inversa de alta eficiencia, con funciones dedicadas a este apartado.
- Generación de PWM que se aproxima a la modulación SVPWM.
- Configuración de los dispositivos periféricos para su lectura o control, como pueden ser los encoders o drivers de transistores que requieren una configuración inicial (AS5047 y DRV8301DCAR).
- Alineación física del encoder con el motor para coordinar el movimiento.
- Cálculo de los intervalos de tiempo discretos entre cada ciclo mediante la función micros(), con esta se mide el tiempo entre ciclos y se usa dicha medida como base para los cálculos discretos de los lazos PI. La librería no usa interrupciones debido a que en ciertos microcontroladores, el uso de interrupciones entra en conflicto con las configuraciones de registros para generar PWM y demás funciones extras de la libreria; además esta opción es utilizada por otros fabricantes de este tipo de controladores, como en ODrive Robotics (2025).

- Aproximación de las funciones seno, coseno , tangente, etc. Las cuales se aproximan de manera que se optimiza su cálculo en microcontroladores sin errores significativos. Esto se debe a que un cálculo extenso de funciones trigonométricas puede bajar la eficiencia en tiempo del controlador.

3.14.1. Estructura del código

En esta sección se describen los puntos más importantes de la librería para la prueba desarrollada en este trabajo de investigación. El código está estructurado en cuatro secciones fundamentales, siguiendo una arquitectura modular para mayor claridad y facilidad de mantenimiento.

1. **Inicialización de periféricos:** Se configura el sensor magnético (AS5047), el controlador de potencia de los transistores (DRV8301DCAR), el motor y los parámetros eléctricos, garantizando que todos los módulos estén listos antes de ejecutar el control.

2. **Configuración del control:** En esta etapa se define el tipo de control (posición, velocidad o par), los controladores PID, además de los límites de voltaje y corriente del motor. Además, se definen los valores constantes del controlador, como las frecuencias de corte de los filtros paso bajo internos. Estos parámetros permiten ajustar el desempeño del sistema y garantizar una respuesta estable frente a perturbaciones.

3. **Bucle principal de control:** Es la parte que se ejecuta continuamente; en ella se calculan las transformaciones del campo (Clarke y Park), se determina el error angular, se aplican los controladores PID y se actualizan las señales PWM. Esta configuración es crítica, ya que define la estabilidad y precisión del sistema. Además, se implementa la modulación SVPWM (Space Vector Pulse Width Modulation), la cual se encarga de generar las señales de conmutación del inversor, optimizando el uso del voltaje del bus DC y reduciendo la distorsión armónica en la salida.

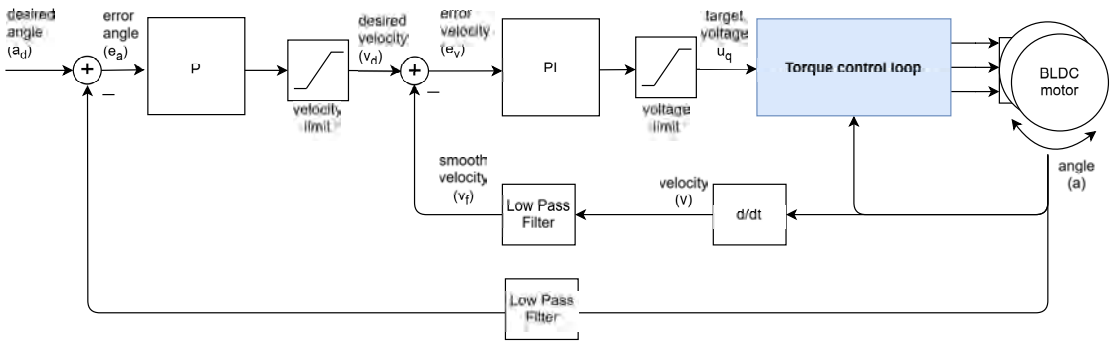
4. **Función agregada para optimización:** Además del código anterior, que es intrínseco de la librería SimpleFOC, se añadieron zonas adicionales en el ciclo que permiten la

realización de la optimización y evaluación de parámetros. Esta parte no es original de la librería, sino fue implementada para la presente investigación.

Los bucles de control, las variables censadas, además de los filtros paso bajo, se comportan de acuerdo al diagrama de bloques del control de posición de la figura 3.25.

Figura 3.25

Diagrama de bloques simpleFoc

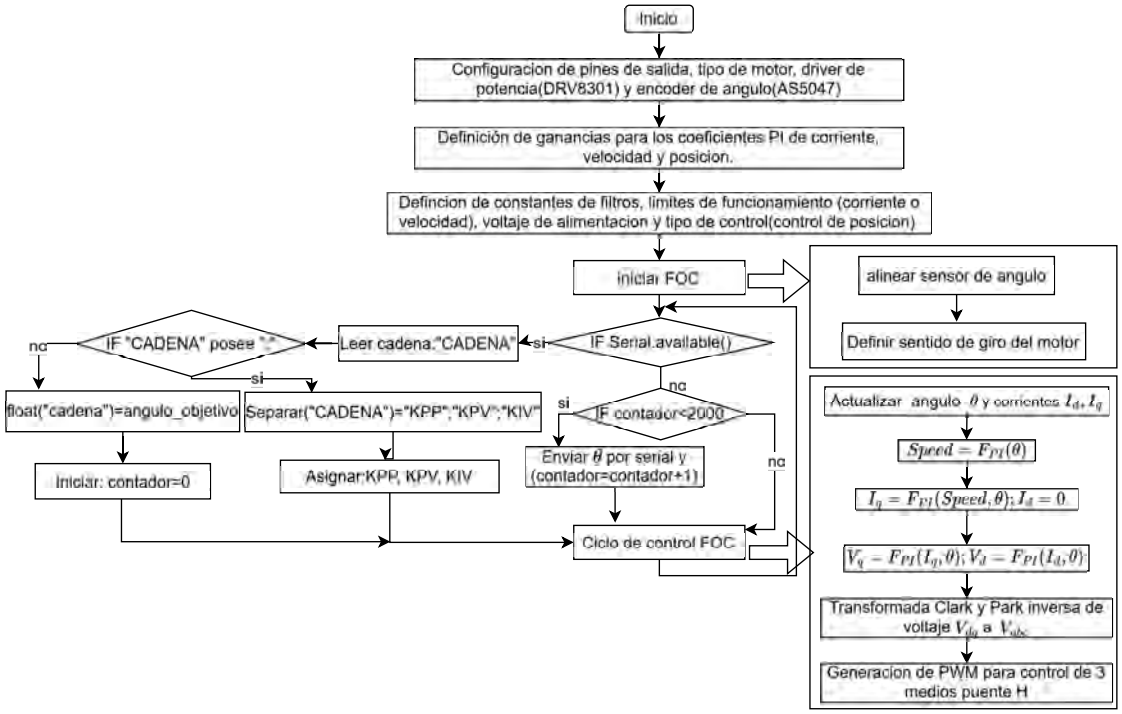


Nota. *Adaptado de Skuric et al. (2022)*

El funcionamiento y estructura del código general son descritos a grandes rasgos por el diagrama de flujo en la figura 3.26.

Figura 3.26

Diagrama de flujo del código del microcontrolador



3.15. Criterios para la optimización del sistema

Para realizar la optimización de los controladores del sistema se utilizó un programa externo al del microcontrolador, el cual fue ejecutado en una PC en paralelo al funcionamiento del controlador. Este programa se desarrolló en el lenguaje Python en el IDE PyCharm y realizó las siguientes tareas.

- Ejecutar los algoritmos de enjambre de partículas o evolución diferencial, según sea la optimización.
- Almacenar los datos y procesarlos según los algoritmos antes mencionados, y generar nuevos coeficientes proporcional P de posición y PI de velocidad, que serán enviados al microcontrolador para realizar pruebas.
- Mostrar los resultados de las pruebas de optimización mediante gráficos y diagramas que muestren los comportamientos de los algoritmos.

3.15.1. Estructura del código de optimización

Este código cambia dependiendo de si se usa el algoritmo de enjambre de partículas o evolución diferencial, pero ambos comparten secciones y la estructura general, las cuales se describen a continuación.

1. **Inicialización de variables:** Se inician las variables que definen el comportamiento de los algoritmos, el número de repeticiones y los rangos de búsqueda de parámetros.
2. **Bucle principal de control:** Se realiza un bucle en el cual se generan coeficientes para el análisis de datos, se los envía por el serial y se espera la respuesta del comportamiento de la planta con los nuevos datos.
3. **Gráficas de los resultados:** Se grafican los comportamientos de las partículas y el de la planta con los diferentes coeficientes.

3.16. Algoritmos de optimización por enjambre de partículas (PSO) y evolución diferencial(DE)

Para la optimización mediante los algoritmos PSO y DE se desarrolló un programa en Python encargado de ejecutar dichos algoritmos. Aunque la carga computacional asociada a cada iteración de los algoritmos de optimización es reducida como se observó en la seccion 3.16.2 y 3.16.3, requiriendo 5 y 8 operaciones, así como un almacenamiento mínimo de 32 Kbits y 30 Kbits para los algoritmos PSO y DE, respectivamente, el algoritmo DE presenta de forma inherente un menor costo computacional y un menor requerimiento de memoria en este caso. No obstante, la mayor carga computacional que se busca evitar en el microcontrolador corresponde a las tareas complementarias asociadas al proceso de optimización.

En particular, se pretende evitar que el microcontrolador realice las siguientes tareas:

- El almacenamiento de los datos generados en iteraciones anteriores, ya que para 50 generaciones, 2000 datos por generación y una resolución de 32 bits se requiere, como mínimo, un almacenamiento de 3.2 Mbits.
- El procesamiento de los datos recolectados para detectar y corregir posibles errores ocasionados por ruido en la comunicación con los periféricos.
- La visualización de los datos para su supervisión antes del almacenamiento permanente, con el fin de detectar errores inesperados durante el proceso.
- El cálculo de la función de penalización previo al almacenamiento de los datos, el cual, al aplicarse a 2000 valores recolectados en cada prueba, requiere aproximadamente 4000 operaciones.

Por lo tanto, con el objetivo de minimizar el tiempo del ciclo de control y favorecer un mayor ancho de banda del lazo de corriente, se delegan al ordenador todas las tareas que no requieren ejecutarse en tiempo real. De este modo, el ordenador ejecuta en paralelo el proceso de optimización, mientras que el microcontrolador se limita a transmitir los 2000 datos generados en cada iteración.

Por otro lado estos algoritmos comparten una característica, la cual es que se ejecutan dentro de los límites de un espacio confinado, puesto que si se aplicara el algoritmo en todo el espacio numérico disponible nunca se llegaría a un resultado; por lo tanto, el primer paso es reducir este espacio de búsqueda lo más posible, esto se logró en la sección 4.2 con un análisis del modelo lineal y heurístico de los lazos de control.

3.16.1. Selección de coeficientes de algoritmos de optimización

Para realizar la comparación del desempeño de los algoritmos, se buscó darles unos coeficientes o características equivalentes, pero adaptados a cada uno, para que así las diferencias fueran dadas por el comportamiento y características del algoritmo y no por los coeficientes.

3.16.2. Optimización por enjambre de partículas

$$v_{i,m}^{(t+1)} = wv_{i,m}^{(t)} + c_1 \times rand() \times \left(pbest_{i,m} - x_{i,m}^{(t)}\right) + c_2 \times rand() \times \left(gbest_m - x_{i,m}^{(t)}\right) \quad (3.54)$$

$$x_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad (3.55)$$

Para este algoritmo, los factores que se pueden escoger son los coeficientes c1 y c2, además del coeficiente de inercia w. El coeficiente C1 influencia la magnitud de la búsqueda del mejor comportamiento local, mientras que el coeficiente C2 influencia la magnitud de la búsqueda del mejor comportamiento global; el coeficiente de inercia afecta la magnitud del ratio de convergencia general de todas las partículas en prueba. Estos coeficientes tienen valores comunes.

- w suele ir de entre 0.4 y 0.9, según se apreció en los trabajos Mustafa and Hashim (2020) y Mukhtar et al. (2019), donde se usa en ambos casos el mismo rango para realizar pruebas comparativas y de funcionamiento.

- C1 y C2 tienen un valor positivo, el cual suele estar restringido en un rango de 0 a 4 como en el trabajo Guong et al. (2018).

El consumo de recursos del algoritmo PSO se puede resumir considerando las operaciones aritméticas por dimensión y los requerimientos de almacenamiento para una configuración de 100 partículas, 50 iteraciones y 3 variables de optimización. La tabla 3.4 presenta un resumen de estos recursos.

Tabla 3.4
Resumen de recursos computacionales del algoritmo PSO.

Recurso	**Cantidad**
Operaciones por dimensión	8
Accesos a vectores por dimensión	3
Partículas consideradas	100
Variables por partícula	3
Iteraciones	50
Valores almacenados	1004
Tamaño por valor	32 bits
Memoria total	32128 bits (4016 bytes)

3.16.3. Optimización por evolución diferencial

Para el algoritmo de evolución diferencial clásico se tienen dos variables que se pueden editar.

$$Y_i^t = X_{r_1}^t + F\left(X_{r_2}^t - X_{r_3}^t\right) \tag{3.56}$$

$$Z_{i,j}^t = \begin{cases} Y_{i,j}^t, & \text{if } rand_{i,j}[0,1] \leq CR \text{ or } j = k \\ X_{i,j}^t, & \text{otherwise} \end{cases} \tag{3.57}$$

- F es el factor de escalado de la mutación. Este factor determina qué tanto afecta el proceso de mutación, siendo que cuanto más grande es, más influencia tiene. Tiene un

valor entre 0 y 1 según Ahmad et al. (2021), pero llega a extenderse de 0 a 2 según el trabajo original del algoritmo de evolución diferencial Storn and Price (1997).

- CR es el coeficiente de cruce de características. Este coeficiente determina la probabilidad de que el nuevo punto herede las características de los donadores y las mutaciones. Tiene un valor entre 0 y 1 según los trabajos Ahmad et al. (2021) y Storn and Price (1997).

Teniendo los valores dentro de los cuales se encuentran los coeficientes, se procede a seleccionar valores que sean relativamente equivalentes para poder realizar las pruebas respectivas; para esto se toma como guía el trabajo Parque and Khalifa (2023), donde se realizan comparaciones de algoritmos de este tipo. En este trabajo se establece que:

- Los coeficientes W y CR tienen una importancia equivalente en lo que refiere a la dirección y magnitud de búsqueda general, por lo cual se utilizaron valores iguales; además, el trabajo mencionado utilizó un valor de 0.5.
- F tiene un valor recomendado de 0.7.
- Este trabajo usó un C1 con un valor de 0.5 y C2 de 2, priorizando la importancia del comportamiento global en lugar del mejor valor local.

Otros valores para los coeficientes han sido recopilados en el trabajo Parque and Khalifa (2023) de muchos casos anteriores, donde comúnmente, de manera heurística o por prueba y error, se llegó a esos parámetros comunes; esto no significa que su funcionamiento sea universal, por lo cual se cambian a conveniencia, siguiendo los límites de valores establecidos en las secciones 4.6.1 y 4.6.2.

Se realizó una prueba previa con los valores recomendados y otros extras para observar su efecto en el comportamiento de los algoritmos y, a partir de los resultados, se seleccionaron los parámetros para las pruebas finales. Se toman como base estos valores por ser muy comunes en trabajos de similar naturaleza al actual, como, por ejemplo, el valor 0.7 para F, el cual se observa en el trabajo Ahmad et al. (2021), que recopila valores de otros trabajos e investigaciones a lo largo del tiempo.

El consumo de recursos del algoritmo DE se puede resumir considerando las operaciones aritméticas por dimensión y los requerimientos de almacenamiento para una configuración de 100 individuos, 50 iteraciones y 3 variables de optimización. La tabla 3.5 presenta un resumen de estos recursos.

Tabla 3.5
Resumen de recursos computacionales del algoritmo DE.

Recurso	**Cantidad**
Operaciones por dimensión	5
Accesos a vectores por dimensión	4
Individuos considerados	100
Variables por individuo	3
Iteraciones	50
Valores almacenados	930
Tamaño por valor	32 bits
Memoria total	29760 bits (3720 bytes)

3.16.4. Función de optimización

Las funciones de optimización determinan el criterio bajo el cual se ejecutan los programas, siendo estas las que se tienen como objetivo reducir a su menor valor mediante las iteraciones. Las más comunes para sistemas de control son:

$$IAE(x) = \int_0^{T_s} |e(t)|, dt \tag{3.58}$$

$$ITAE(x) = \int_0^{T_s} t|e(t)|, dt \tag{3.59}$$

$$ITSE(x) = \int_0^{T_s} te(t)^2, dt \tag{3.60}$$

$$ISE(x) = \int_0^{T_s} e(t)^2, dt \tag{3.61}$$

En este trabajo se seleccionó el índice ITAE debido a que proporciona una evaluación

representativa del comportamiento transitorio y del desempeño final del sistema de control de posición. De acuerdo con Dorf (2011), este índice fue propuesto para reducir la influencia del elevado error inicial sobre la integral de desempeño y, al mismo tiempo, incrementar la ponderación de los errores que permanecen en etapas posteriores de la respuesta. De esta manera, el factor temporal permite penalizar con mayor intensidad los errores persistentes, mientras que disminuye la relevancia de los errores iniciales asociados a cambios de referencia, los cuales son inevitables durante la respuesta transitoria del sistema.

Estas características hacen del ITAE una función objetivo adecuada para la optimización de un sistema de control de posición de un motor BLDC, por lo que la minimización de este índice promueve las siguientes características:

- Menor tiempo de establecimiento.
- Reducción de las oscilaciones y vibraciones.
- Disminución del error en estado estacionario.
- Mayor precisión en el seguimiento de la referencia.

Asimismo, Luo et al. (2022) señalan que el ITAE constituye una función de evaluación adecuada para lograr un buen rendimiento de control y reducir el tiempo de ajuste de un motor sin escobillas síncrono de imán permanente. Además, su efectividad como criterio individual de optimización se encuentra respaldada por investigaciones donde se emplea sin combinarlo con otros parámetros de desempeño, como los trabajos reportados en Zhao et al. (2021) o Mustafa and Hashim (2020).

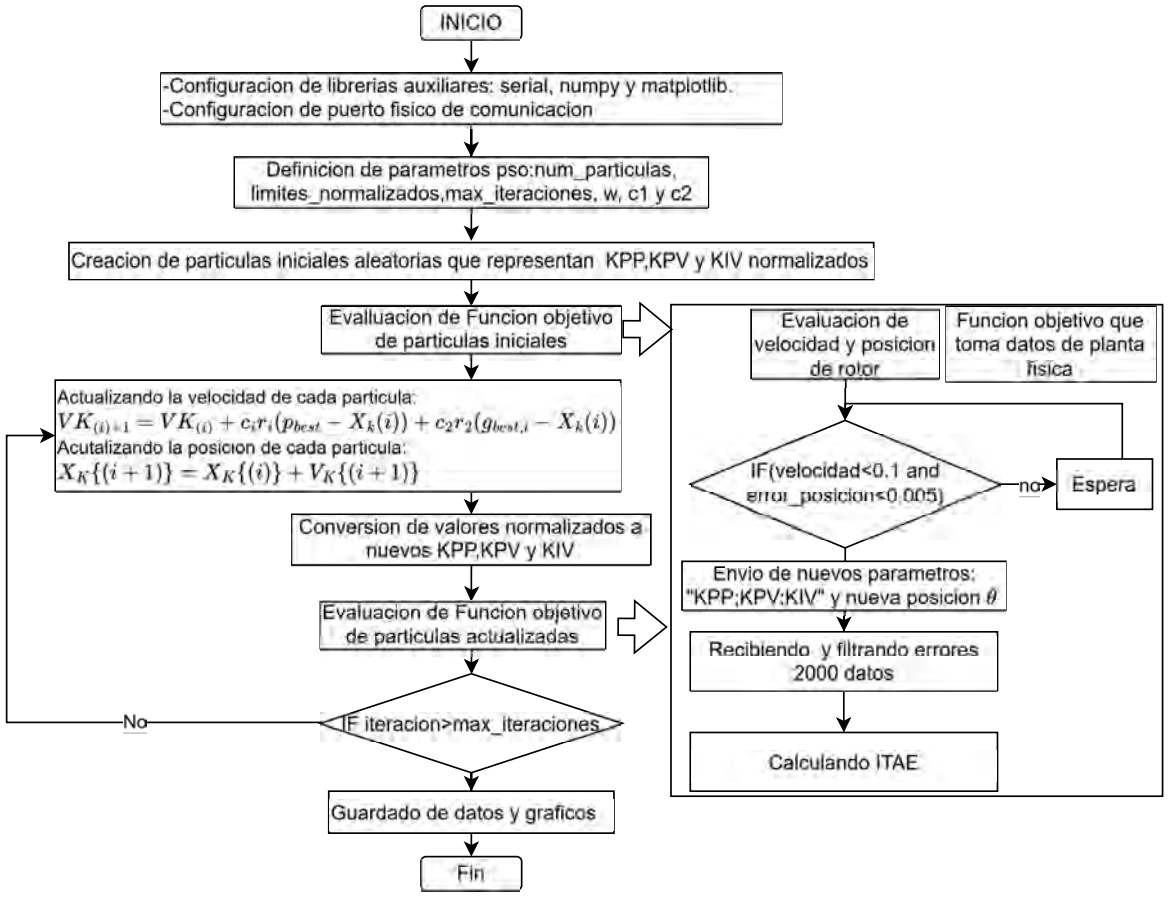
Por último, si bien las funciones objetivo compuestas permiten considerar múltiples características de desempeño, estas requieren la definición de coeficientes de ponderación cuya selección depende de los objetivos específicos del sistema. Debido a que en la documentación revisada no se encontró un criterio general para establecer dichos valores, se optó por utilizar el ITAE como función objetivo única, evitando incorporar criterios adicionales de ajuste y manteniendo un proceso de optimización más simple y consistente.

3.16.5. Diagrama de flujo de los algoritmos usados

Para el algoritmo de enjambre de partículas

El código de optimización se desarrolló en Python y está basado fundamentalmente en las ecuaciones que se mostraron en la sección 2.2.7 y sigue el diagrama de flujo de la figura 3.27. Su implementación comprende la inicialización de las partículas, la evaluación de la función objetivo y la actualización iterativa de sus posiciones y velocidades de acuerdo con las ecuaciones del algoritmo. En este proceso, el trabajo realizado por el microcontrolador representa la evaluación de la función objetivo, ya que ejecuta el controlador con los parámetros propuestos por el algoritmo y devuelve los valores de desempeño que serán utilizados durante la optimización. Además, en cada iteración se realizan comprobaciones y filtrado de datos para detectar posibles errores no representados en el diagrama de flujo.

Figura 3.27
Diagrama de flujo del algoritmo PSO

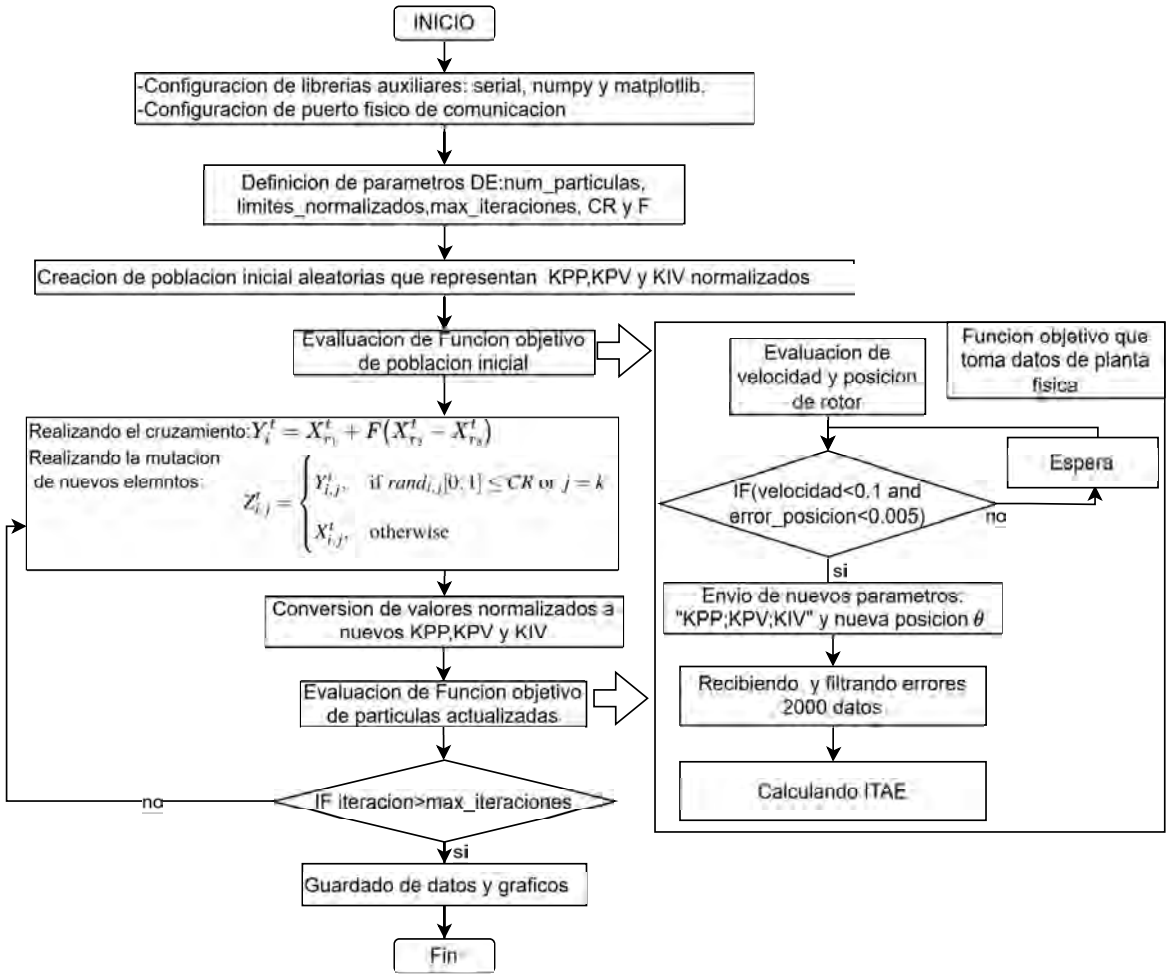


Para algoritmo de evolución diferencial

Por otro lado, para el algoritmo de Evolución Diferencial, el código está basado en la figura 2.15, donde se muestra un pseudocódigo de esta técnica, del cual se extrae el diagrama de flujo de la figura 3.28 y se construye el código a partir de ese punto. Al igual que en el algoritmo PSO, la evaluación de la función objetivo es realizada por el microcontrolador, el cual ejecuta el controlador con los parámetros generados por cada individuo y devuelve los valores de desempeño necesarios para el proceso de optimización. Asimismo, durante cada iteración se realizan comprobaciones y filtrado de datos para detectar posibles errores no representados en el diagrama de flujo. Se utilizó la misma función objetivo que se empleó en el algoritmo anterior.

Figura 3.28

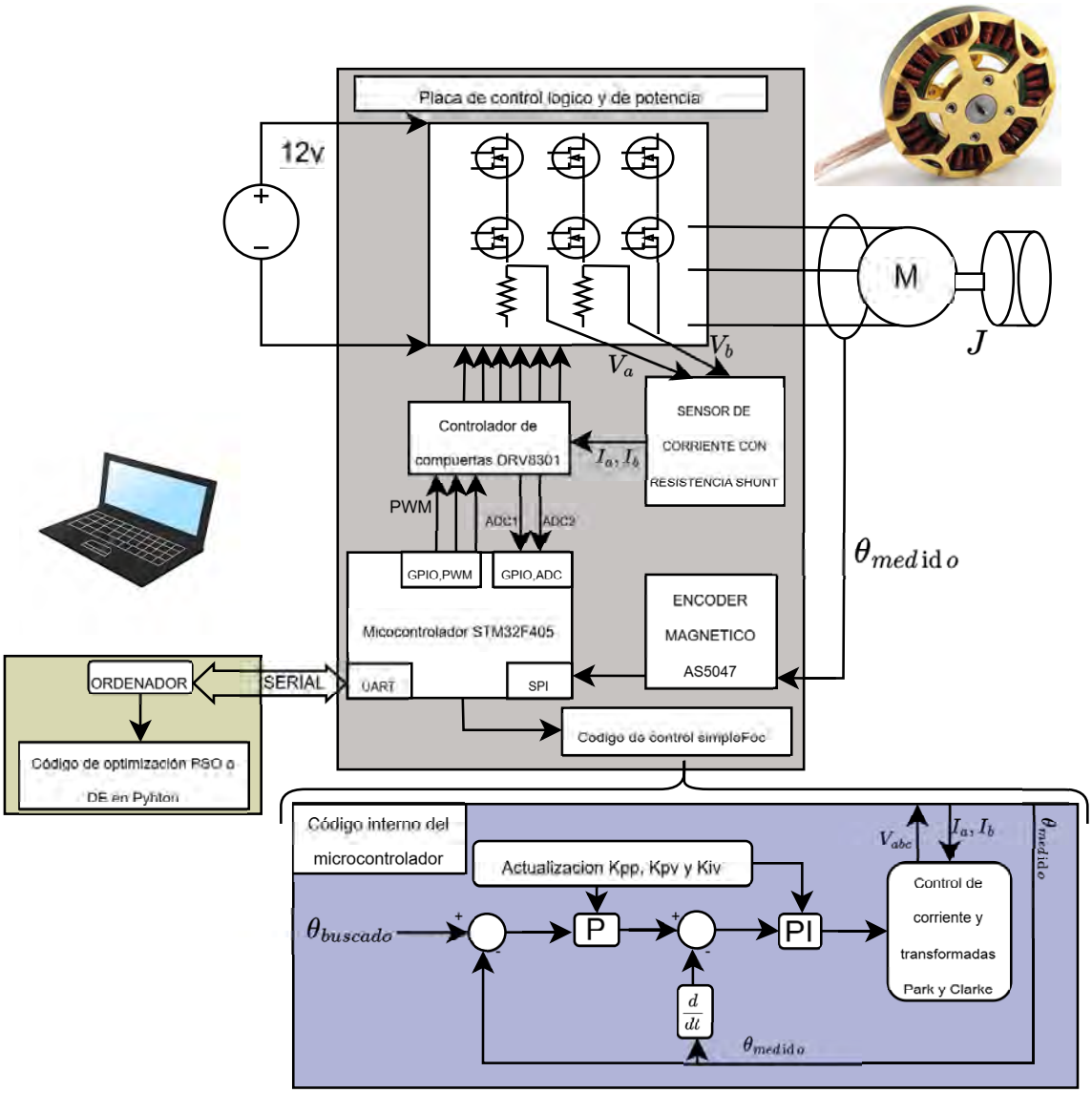
Diagrama de flujo del algoritmo DE



3.17. Diagrama general del sistema implementado

En esta sección se muestra el diagrama general del sistema, donde se representan la placa de control de potencia y lógica, el ordenador encargado de ejecutar los algoritmos de optimización y el conjunto motor–encoder con la carga mecánica, como se observa en la figura 3.29. Estos elementos conforman un sistema de optimización en lazo cerrado, en el que el ordenador envía los parámetros al microcontrolador y, a partir de la respuesta medida por el encoder, calcula la función de coste para generar nuevas soluciones hasta finalizar la optimización.

Figura 3.29
Diagrama general del sistema implementado



CAPÍTULO 4:

RESULTADOS Y DISCUSIÓN

4.1. Características de pruebas a realizar

Las pruebas de optimización se aplicaron a los coeficientes de los lazos de velocidad y posición del controlador, incluyeron las siguientes características generales y particulares, además de ciertas pruebas iniciales para adecuar los algoritmos de optimización a la función objetivo planteada.

- La prueba fue de tipo respuesta escalón de posición con magnitud 3.14 rad o 180 grados sexagesimales.
- Se utilizó la función ITAE para la caracterización de resultados de las pruebas.
- Se utilizó un cierto número de partículas o pruebas iniciales, como 30, 50 o 100, que también sirvieron para comparar comportamientos.
- Se realizaron pruebas iniciales para calibrar las características de los algoritmos que no son similares entre sí, como los coeficientes c1 y c2 para el algoritmo PSO.

4.2. Pruebas iniciales de calibración

Para las pruebas iniciales con los algoritmos, se utilizaron 5 partículas en 10 iteraciones para reducir el tiempo por cada prueba y comparar los efectos de los parámetros que no son compartidos por ambos algoritmos.

Los valores de coeficientes base para las búsquedas calculados previamente son:

$$Kpi = 0.116; Kii = 111; Kpv = 0 - 34.1; Kiv = 0 - 828; Kpp = 0 - 5 \tag{4.1}$$

Todos fueron multiplicados por un valor arbitrario de 1.5 para asegurar que se cubra un rango que incluya todos los parámetros que permiten que el sistema sea estable. Este valor puede cambiar en caso de observarse que el mejor comportamiento se dé tendiente a un punto fuera del rango actual o de que el rango sea muy extenso.

4.2.1. Pruebas de optimización por enjambre de partículas(PSO) y evolución diferencial(DE)

PSO

Para este algoritmo se cambiaron los parámetros de C1 y C2, que tienen un rango de 0 a 2, y se observó su comportamiento en las iteraciones. El valor del parámetro W se fijó en valores comunes y extremos, como son el valor recomendado 0.5 y el 1, para apreciar el efecto de este valor, que afecta el comportamiento general del algoritmo.

Las figuras mostradas representan el comportamiento de las partículas o de los valores de población actual a través de las iteraciones o generaciones, mostrando como referencia el valor de posición de cada partícula y un rastro de su movimiento. La visualización se realiza en tres planos en bidimensionales para una mayor claridad en las ubicaciones de cada partícula, ya que en un gráfico 3D se dificulta más la asociación de una partícula y su valor en cada eje. Esto se aprecia en las figuras 4.1,4.2, 4.3 y 4.4. Los valores de cada dimensión están normalizados con respecto al rango del coeficiente respectivo que representan.

$$Dimension1 = \frac{P_{posicion}}{Rango(P_{posicion})} = Kpp_{normalizado} \tag{4.2}$$

$$Dimension2 = \frac{P_{veloicidad}}{Rango(P_{veloicidad})} = Kpi_{normalizado} \tag{4.3}$$

$$Dimension3 = \frac{I_{veloicidad}}{Rango(I_{veloicidad})} = Kvi_{normalizado} \tag{4.4}$$

Figura 4.1

Trayectorias de las partículas W=0.5, c1=0.5 y c2=0.5

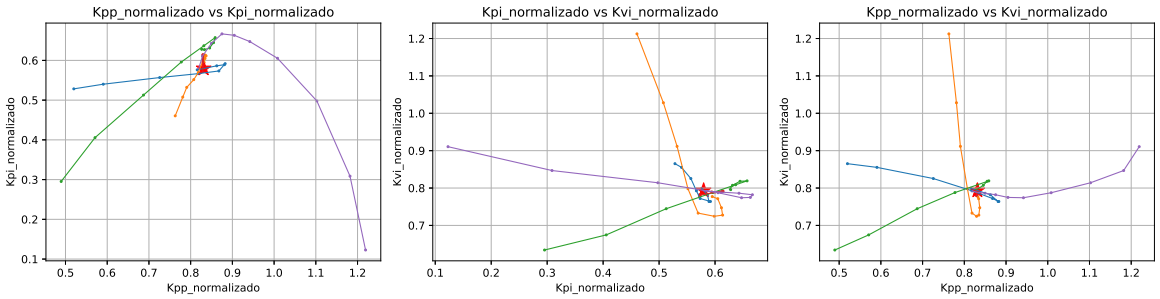


Figura 4.2

Trayectorias de las partículas W=0.5, c1=0.5 y c2=2

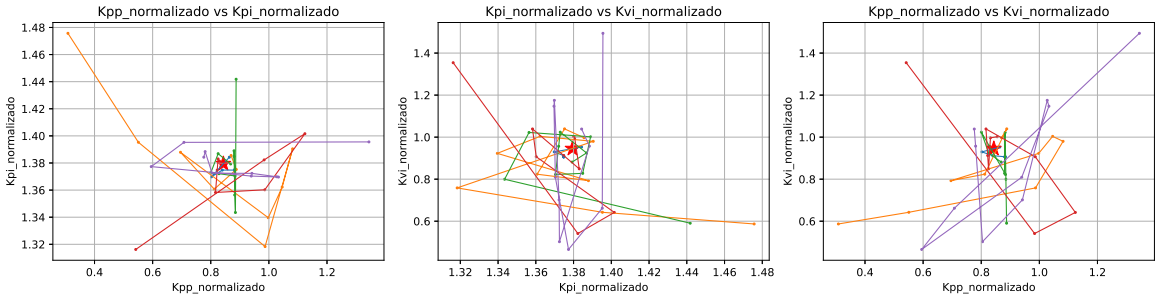


Figura 4.3

Trayectorias de las partículas W=0.5, c1=2 y c2=0.5

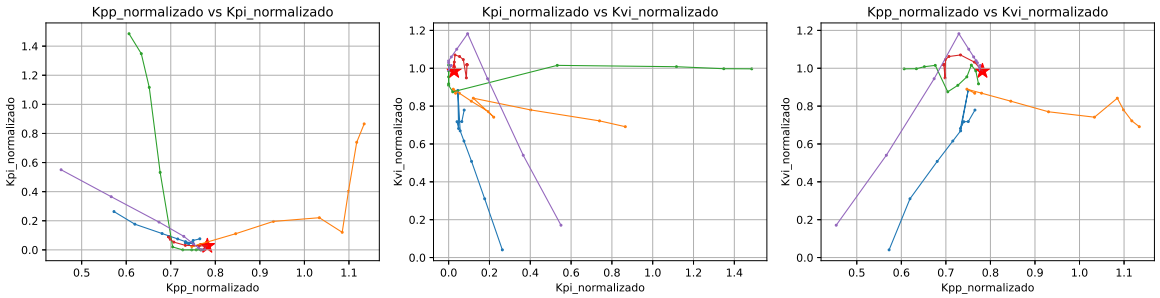
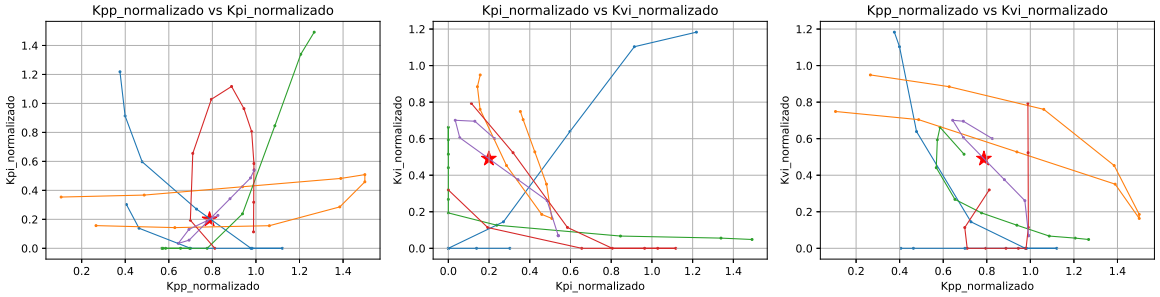


Figura 4.4

Trayectorias de las partículas W=1, c1=0.5 y c2=0.5



Se observaron los gráficos y se optó por los siguientes valores para el algoritmo PSO:

$$W = 0.5,\ C1 = 2,\ C2 = 0.5 \tag{4.5}$$

Se seleccionaron estos valores priorizando la búsqueda individual de cada partícula en lugar de la del grupo, puesto que todas las pruebas mostraron efectividad dirigiéndose al punto de mayor puntaje; mas al tomar rutas demasiado directas o tener una dirección muy limitada hacia el mejor puntaje, no exploran en su camino en busca de posibles soluciones, aspecto que el algoritmo con los factores seleccionados realiza de mejor manera, como se muestra en la figura 4.3.

Pruebas de optimización por evolución diferencial(DE)

Para esta sección se cambiaron los parámetros de F y CR, que tienen un rango de 0 a 2 y 0 a 1 respectivamente, y luego se observó su comportamiento en las iteraciones. Para mostrar el comportamiento, se utilizaron los parámetros en sus extremos como se muestra en las figuras 4.5, 4.6, 4.7, 4.8, 4.9 y 4.10.

Figura 4.5

Trayectorias de las partículas F=0.5 y CR=1

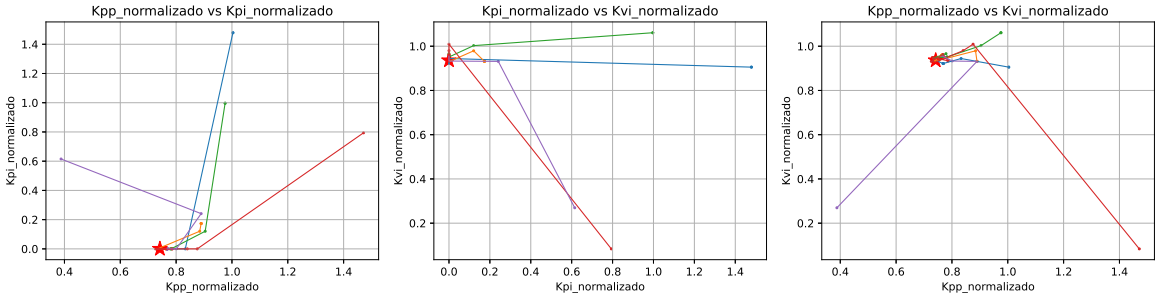


Figura 4.6

Trayectorias de las partículas F=1 y CR=0.5

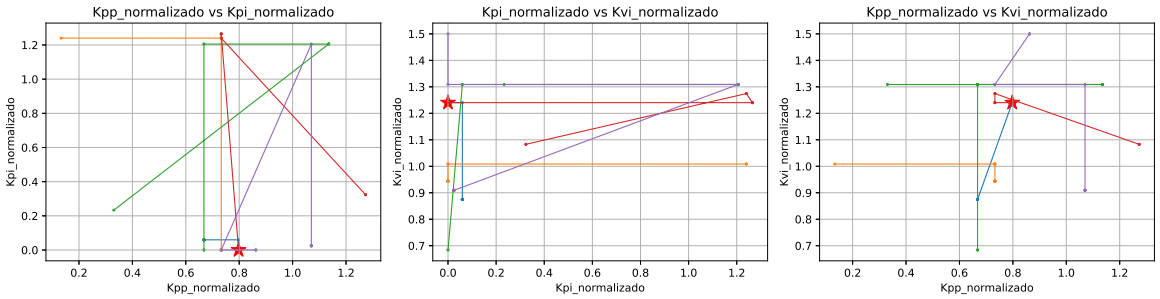


Figura 4.7

Trayectorias de las partículas F=0.5 y CR=0.5

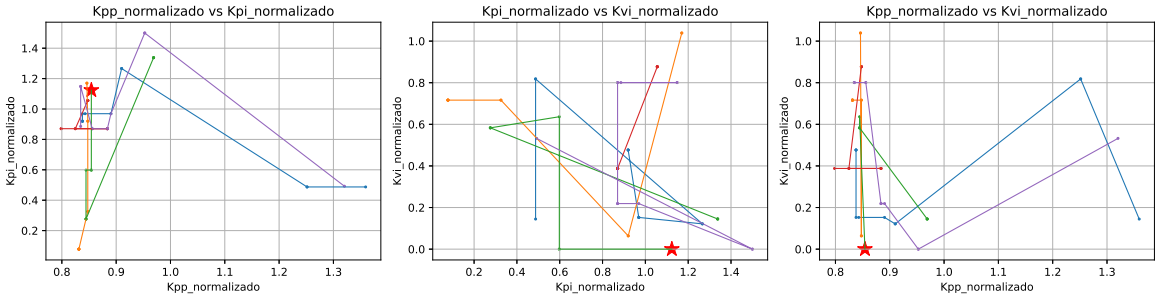


Figura 4.8

Trayectorias de las partículas F=0.7 y CR=0.5

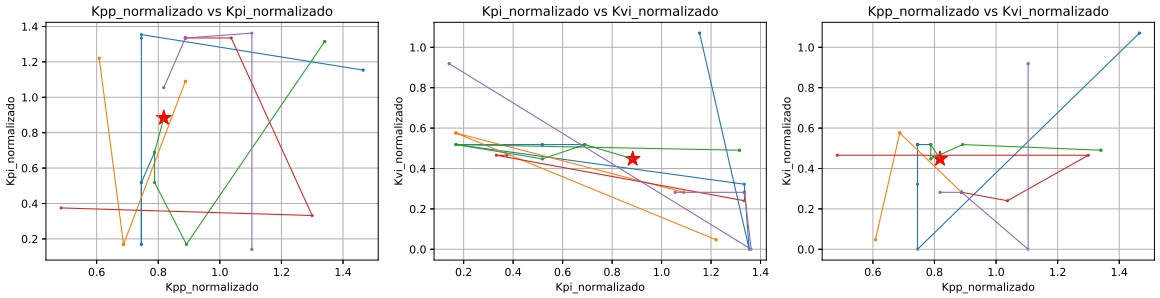
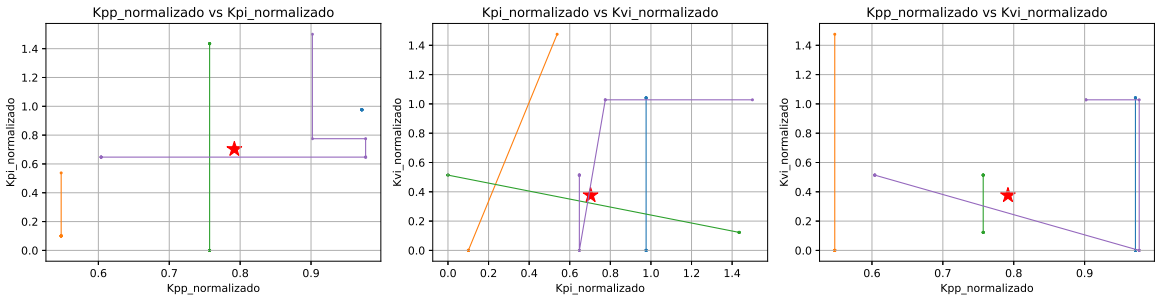


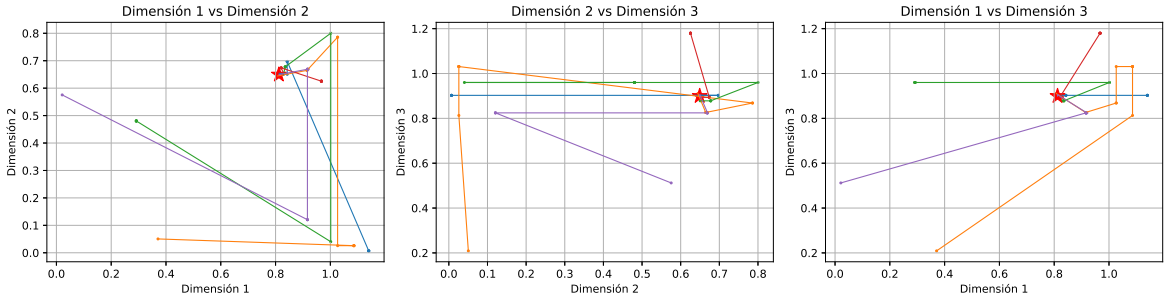
Figura 4.9

Trayectorias de las partículas F=2 y CR=0.5



Para el caso del algoritmo de evolución diferencial, se tomaron los factores:

Figura 4.10
Trayectorias de las partículas F=0.2 y CR=0.5



$$CR = 0.5; F = 0.2 \tag{4.6}$$

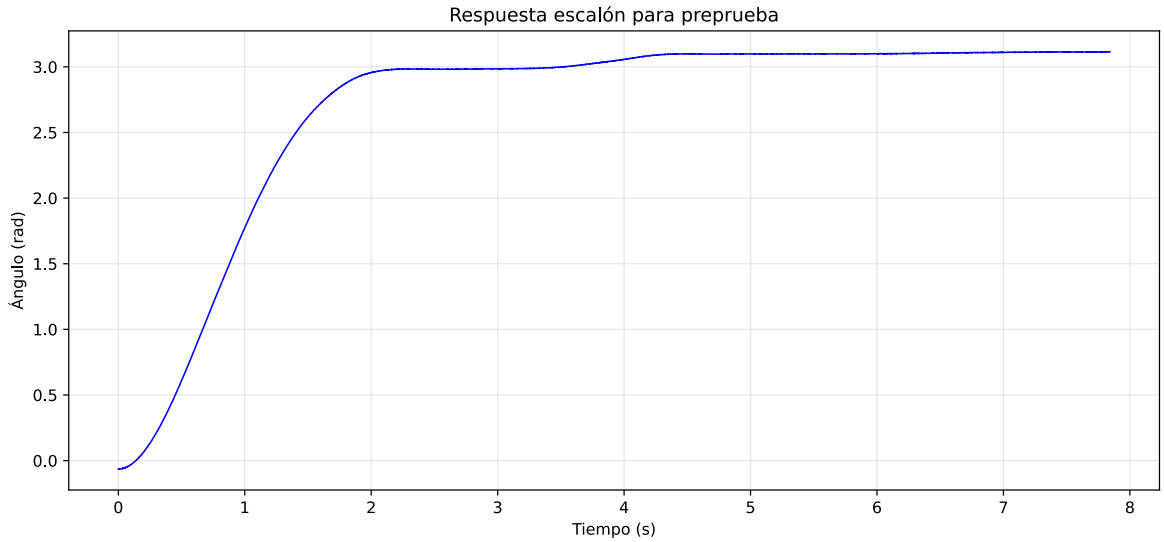
La selección de estos valores se realizó debido a que el valor de CR de 1, muestra una variabilidad muy grande, llegando a los extremos de los rangos de búsqueda con frecuencia; además, tomando en cuenta que este valor es equivalente en efecto al W del algoritmo de PSO, se tomó 0.5 para evitar los saltos demasiado largos de búsqueda y mantener la coherencia con el algoritmo anterior. En el caso de F, se tomó 0.2 puesto que es un valor que procura una búsqueda que no vaya solo de extremo a extremo, como en la figura 4.9, donde las partículas se mantienen en los extremos debido al valor 2 de F. Con esto se buscó procurar el mismo comportamiento base de los dos algoritmos, permitiendo que tenga una búsqueda local no muy excesiva y que esta converja con el tiempo a la zona de mayor puntaje.

4.3. Pruebas Iniciales

Previo a las pruebas extensivas se mostró el resultado de la preprueba planteada para el estudio, esta consistió de la prueba escalón para el controlador con todos los coeficientes en valor unitario, obteniendo el resultado mostrado en la figura 4.11.

En esta prueba se obtuvieron las características de sobreimpulso de 0 % un error de estado estacionario de 0.73 % y un tiempo de establecimiento de 4.16 segundos. Estos resultados fueron ampliamente superados luego de la optimización con excepción del sobreimpulso el cual es 0 porque la preprueba nunca alcanza realmente la referencia.

Figura 4.11
Escalón de 3.14 para preprueba



4.4. Pruebas extensivas

Con los valores seleccionados previamente para los algoritmos, se procedió a las pruebas donde lo único que se cambió fue el número de partículas a utilizar, puesto que se dejo el número de iteraciones en un valor de 50.

Para los algoritmos correspondientes, se realizaron 9 pruebas donde la diferencia entre las mismas fue el número de partículas o población inicial, el cual fue será de 30, 50 y 100.

4.4.1. Prueba del algoritmo PSO con 30, 50 y 100 partículas iniciales

Los gráficos de las pruebas mostradas constan del ángulo de movimiento del rotor hasta llegar al ángulo objetivo de 3.14 rad que se muestra en las figuras 4.12,4.13 y 4.14, además, las pruebas generaron formas de onda parecidas a un escalón de orden 2. En estas se aprecia que para todas las pruebas que el sobreimpulso es muy bajo y que el comportamiento general de las curvas es prácticamente identico.

Figura 4.12

Mejores pruebas escalón de 3.14 rad para 30 partículas

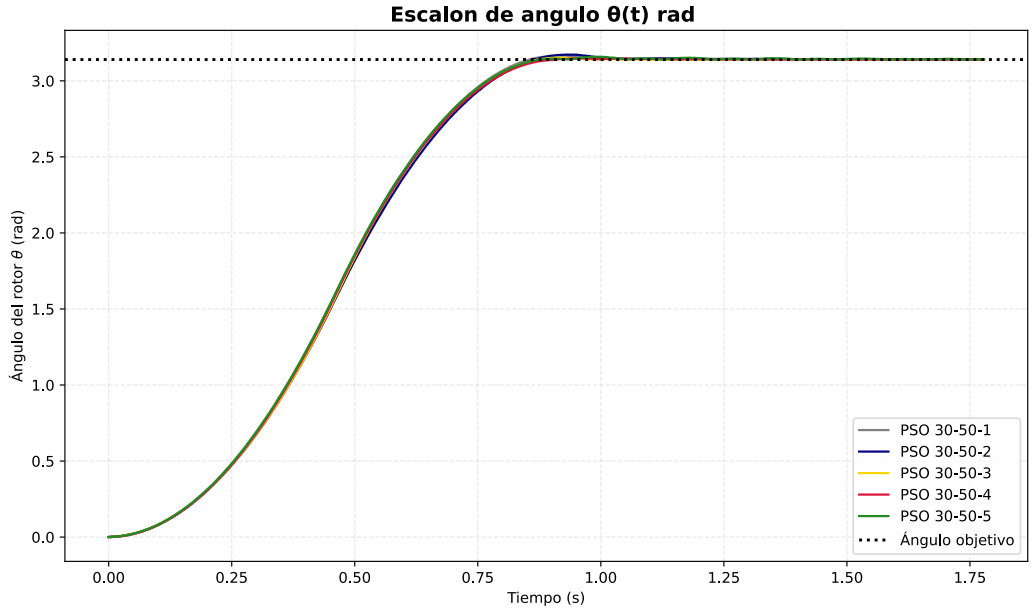
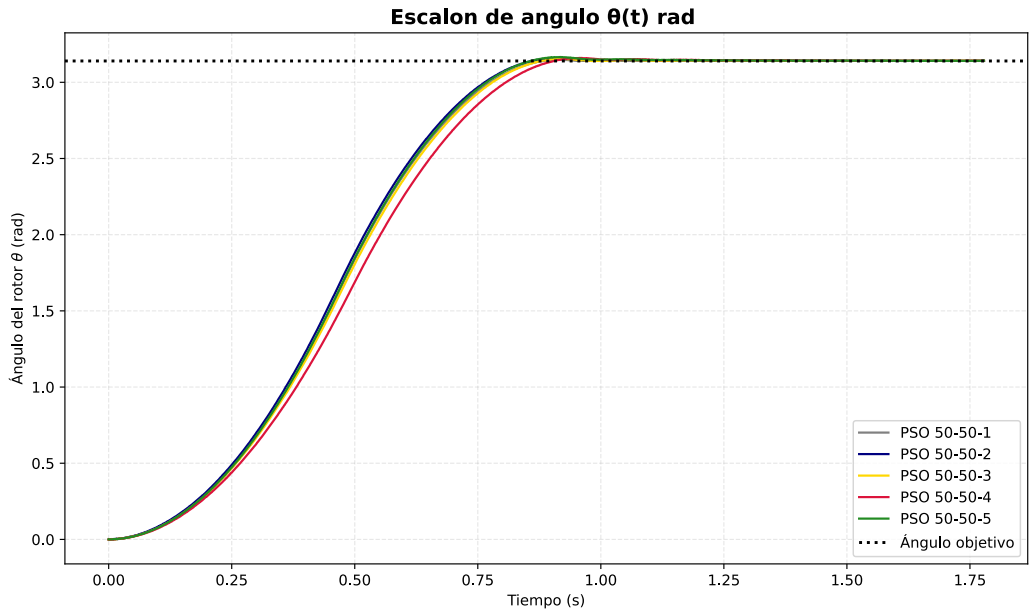


Figura 4.13

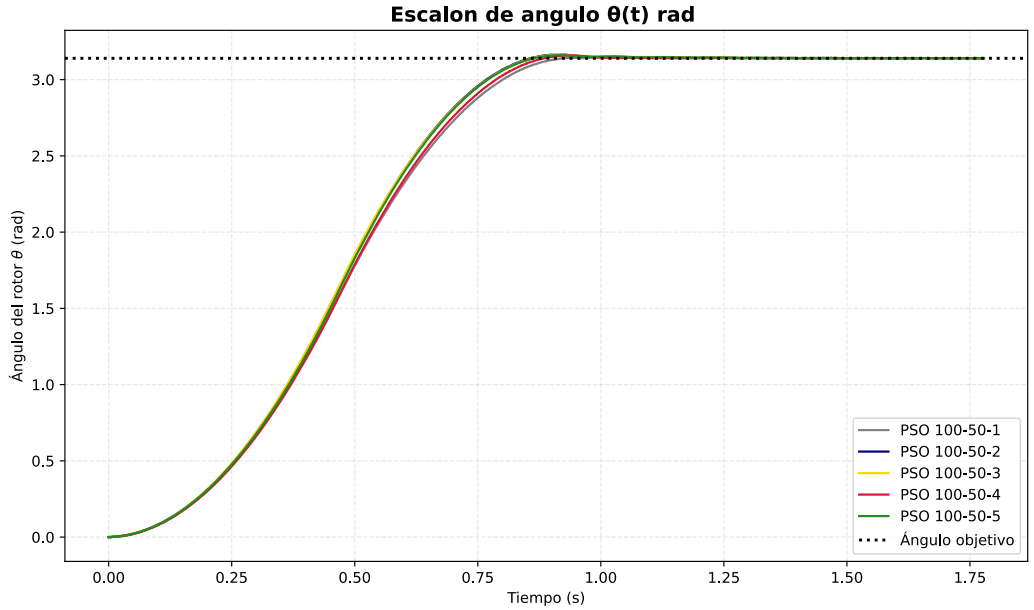
Mejores pruebas escalón de 3.14 rad para 50 partículas



Para hallar el valor de ITAE se calculó la suma del valor absoluto del error en cada punto multiplicado por un valor proporcional al tiempo transcurrido, ademas para nuestro caso se le agregó una constante de 0.05 o una división entre 20, esto solo con el propósito de reducir los puntajes a números no muy extensos. Se obtuvo la curva de puntajes que se irán sumando para el cálculo del ITAE como se muestra en las figuras 4.15,4.16 y 4.17, que

Figura 4.14

Mejores pruebas escalón de 3.14 rad para 100 partículas



siguen la ecuación:

$$\text{ITAE} = \sum_{k=1}^{N} \frac{k}{20} \cdot |e(k)| \tag{4.7}$$

Figura 4.15

Mejores curvas de valor absoluto de error para 30 partículas

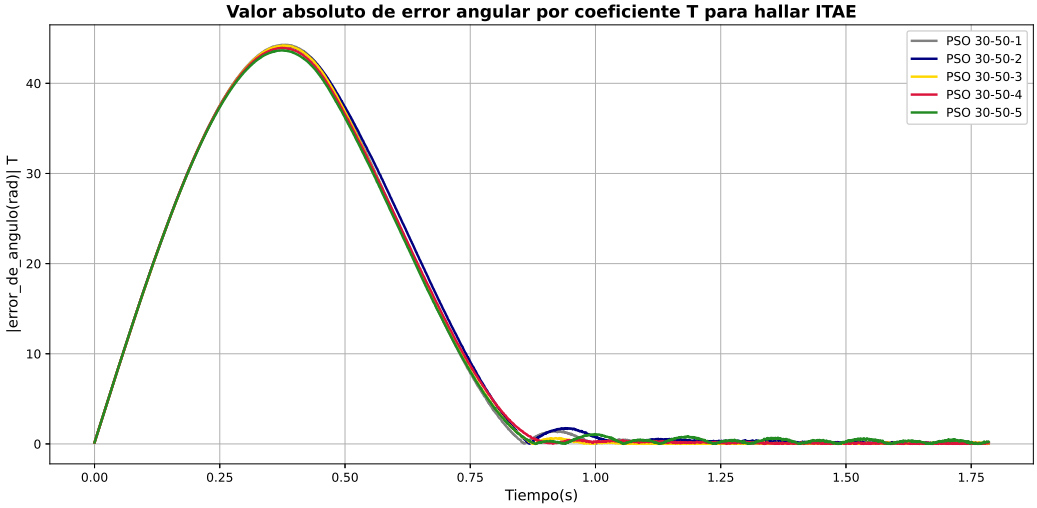


Figura 4.16
Mejores curvas de valor absoluto de error para 50 partículas

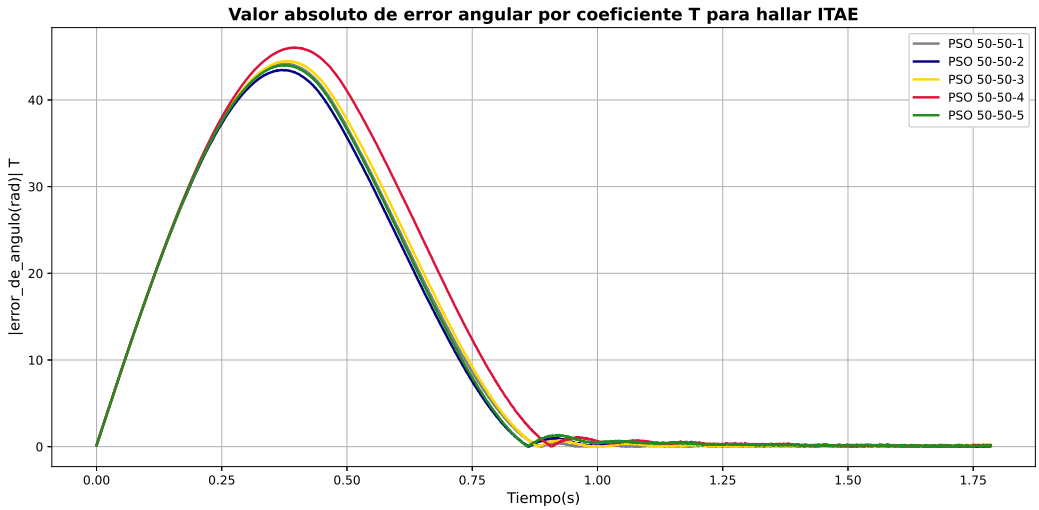
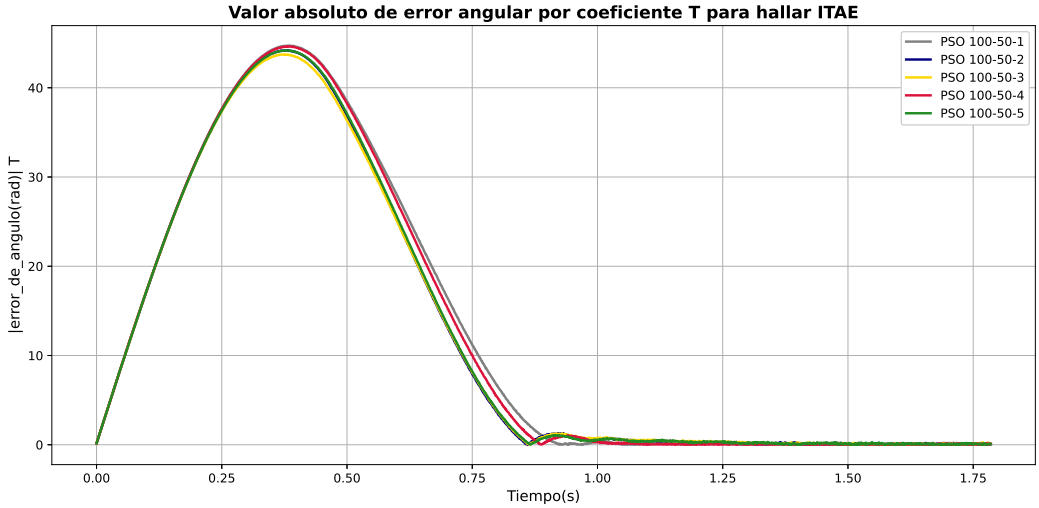


Figura 4.17
Mejores curvas de valor absoluto de error para 100 partículas



También se graficó cómo fueron evolucionando los puntajes a lo largo de la iteraciones. Durante las pruebas se demostró que el menor puntaje fue disminuyendo conforme pasaban las generaciones, todas las pruebas cumplieron con esta característica en mayor o menor medida lo que evidencia que el algoritmo de búsqueda cumplió su propósito esto se aprecia en las figuras 4.18,4.19 y 4.20.

Figura 4.18

Mejores puntajes para 30 partículas

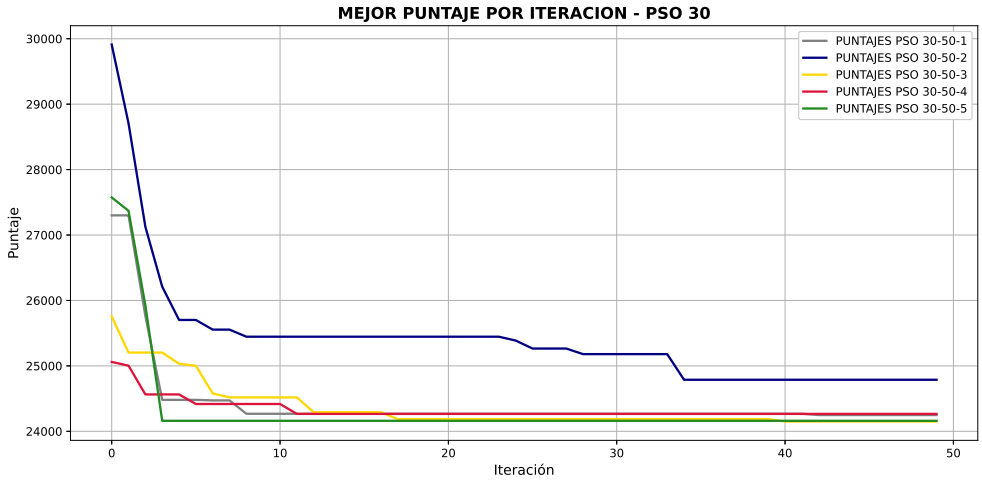


Figura 4.19

Mejores puntajes para 50 partículas

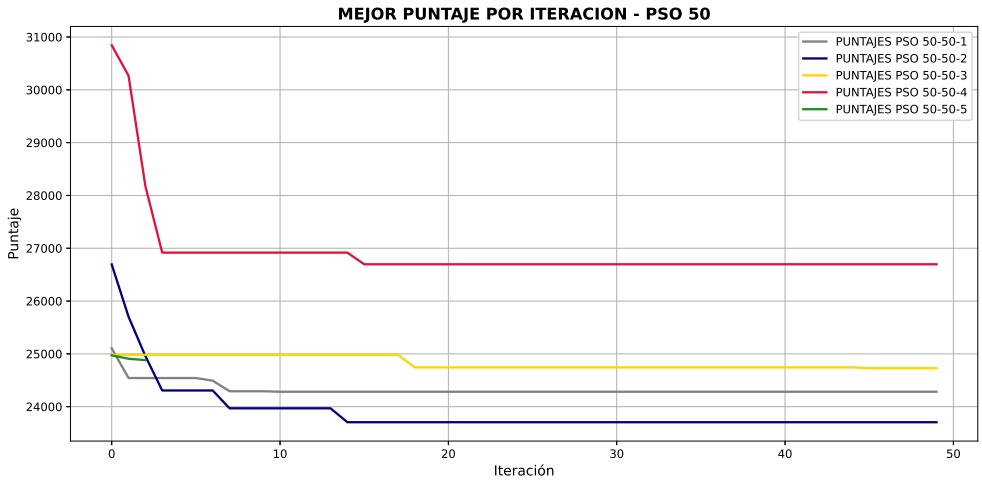
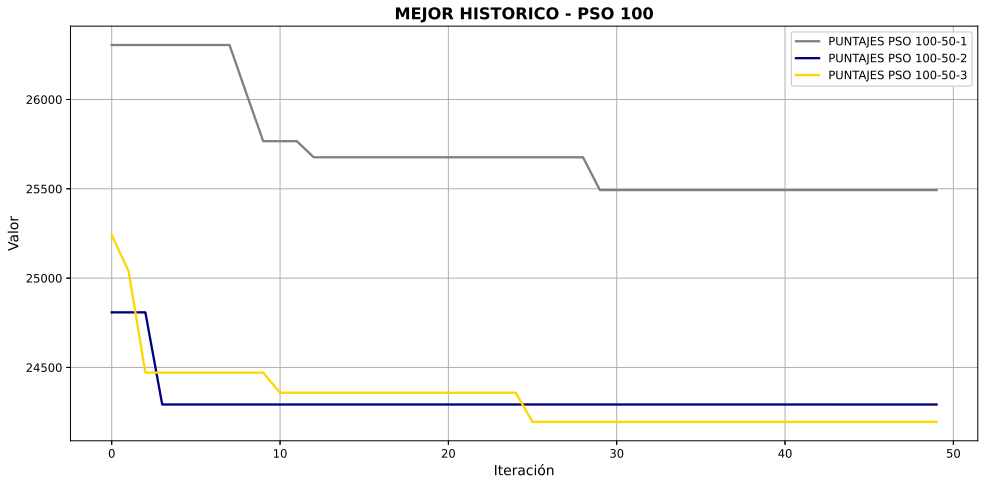


Figura 4.20

Mejores puntajes para 100 partículas

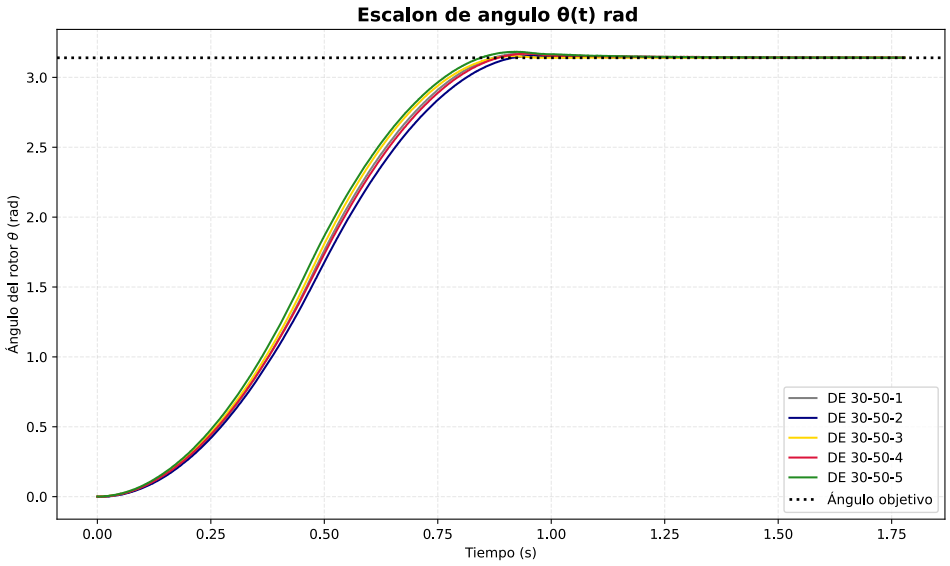


4.4.2. Prueba del algoritmo DE con poblaciones iniciales de 30, 50 y 100 individuos de prueba.

Para el caso del algoritmo de evolución diferencial se realizaron pruebas iguales a las del anterior algoritmo que constan de enviar un escalón al ángulo de movimiento del rotor hasta llegar al ángulo objetivo de 3.14 rad que se muestra en las figuras 4.21,4.22 y 4.23; además las gráficas muestran que el mayor sobreimpulso para una optimización se obtuvo en la primera prueba con una población inicial de 50, mas quitando la excepción el sobreimpulso siguió con la tendencia de ser insignificante.

Figura 4.21

Mejores pruebas escalón de 3.14 rad para 30 partículas



Para hallar el valor de ITAE, del mismo modo que el caso anterior se calculó la suma del valor absoluto del error en cada punto multiplicado por un valor proporcional al tiempo transcurrido, y se le agregó una constante de 0.05 o una división entre 20. Se obtuvieron las curvas de las figuras 4.24,4.25 y 4.26 que siguen la ecuación:

$$\text{ITAE} = \sum_{k=1}^{N} \frac{k}{20} \cdot |e(k)| \tag{4.8}$$

Figura 4.22
Mejores pruebas escalón de 3.14 rad para 50 partículas

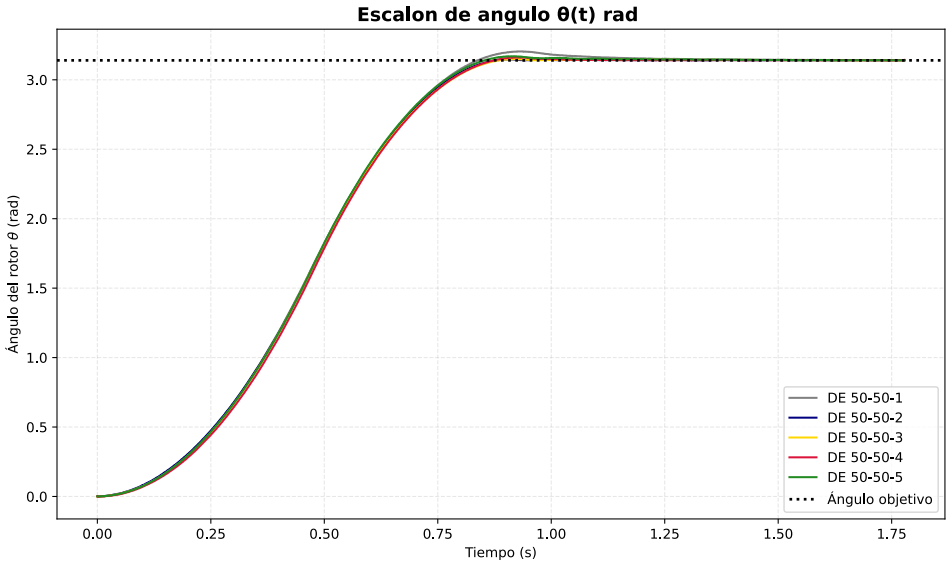
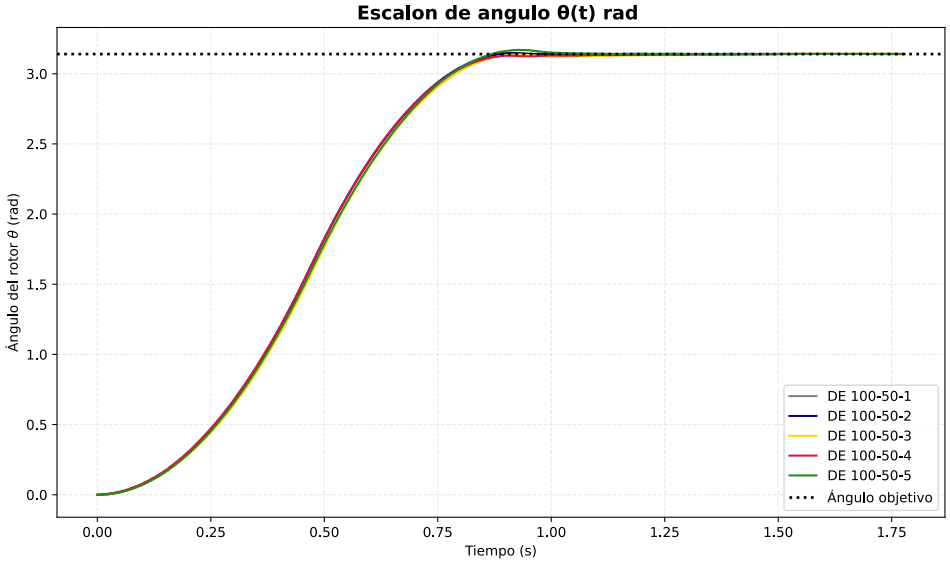


Figura 4.23
Mejores pruebas escalón de 3.14 rad para 100 partículas



Por último se graficó la evolución de los puntajes. En el caso de este algoritmo también se cumplió con la disminución progresiva del puntaje, lo que va acorde a un correcto funcionamiento del algoritmo como se muestra en las figuras 4.27,4.28 y 4.29.

Figura 4.24

Mejores curvas de valor absoluto de error para 30 partículas

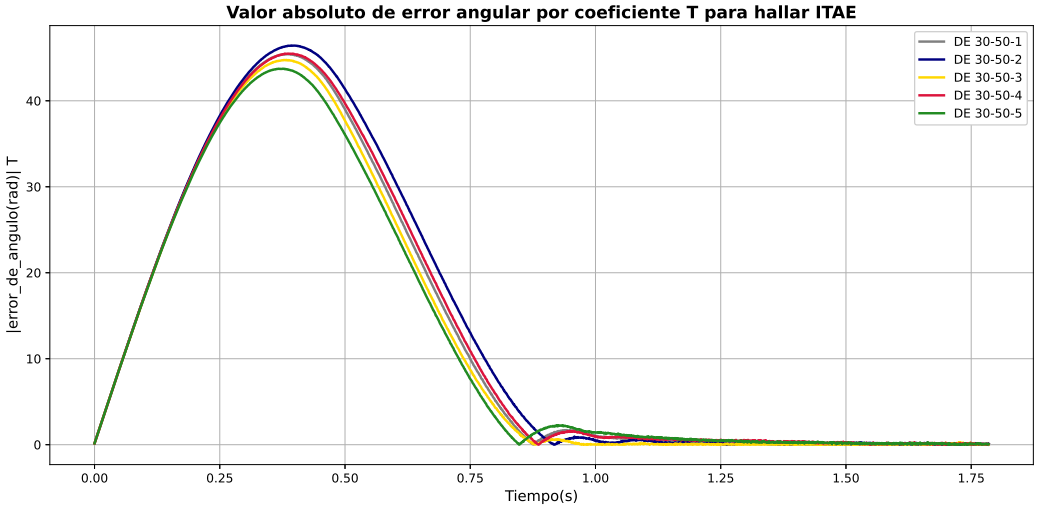


Figura 4.25

Mejores curvas de valor absoluto de error para 50 partículas

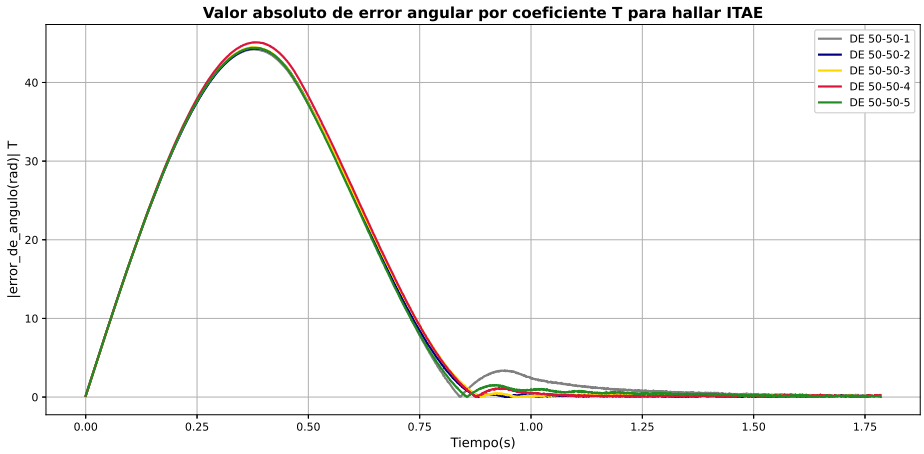


Figura 4.26

Mejores curvas de valor absoluto de error para 100 partículas

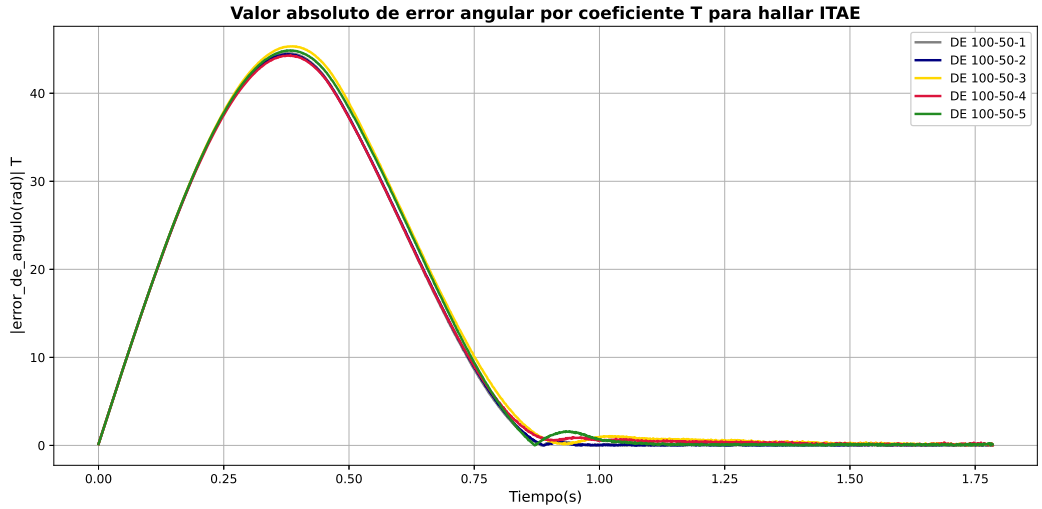


Figura 4.27

Mejores puntajes para 30 partículas

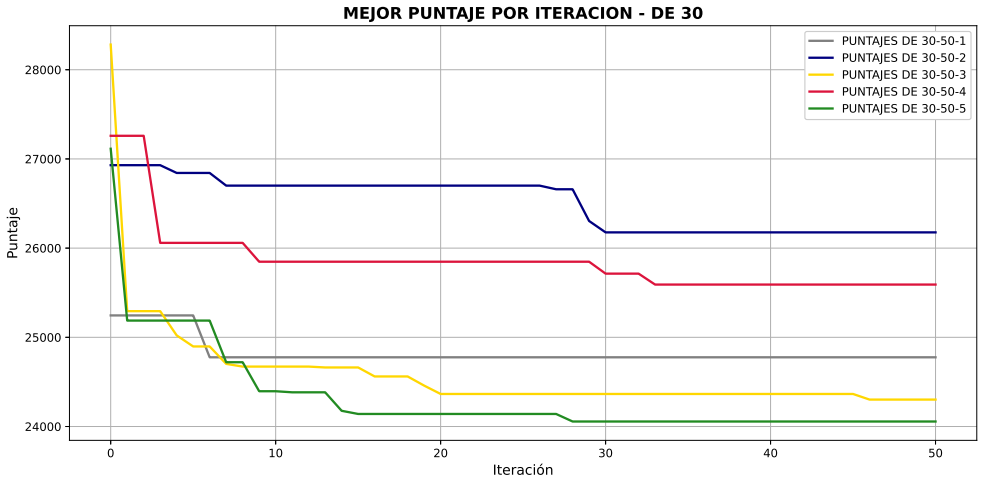


Figura 4.28

Mejores puntajes para 50 partículas

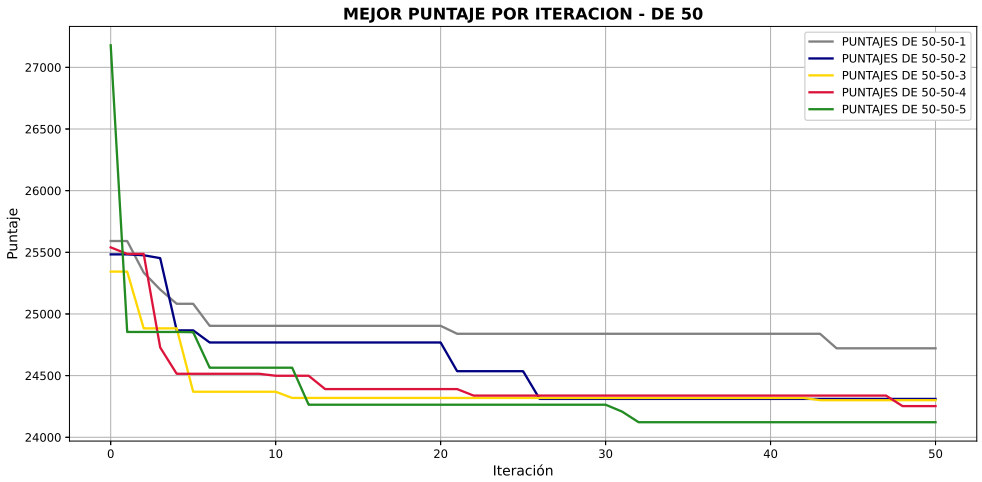
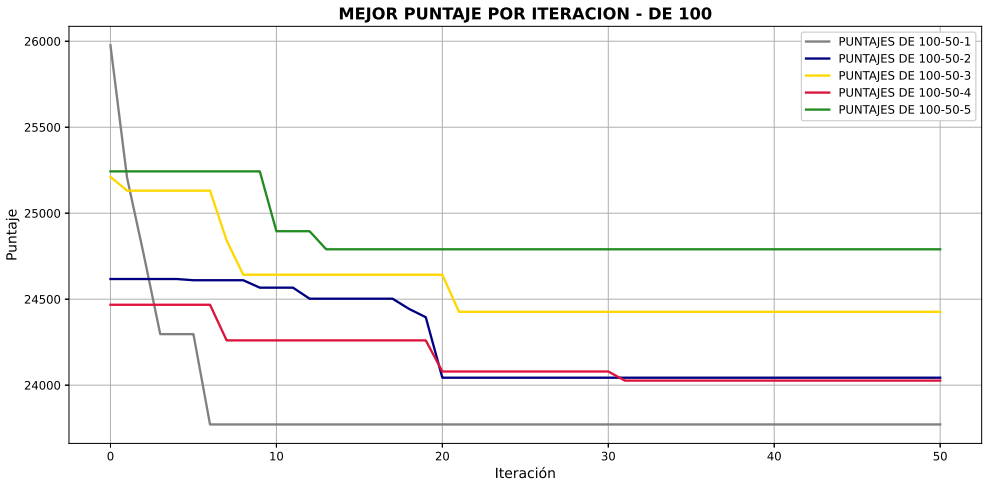


Figura 4.29

Mejores puntajes para 100 partículas



4.4.3. Gráficos comparativos finales de todas las pruebas

Finalmente, se graficaron todas las pruebas juntas, con lo cual se denota una gran uniformidad entre ambos algoritmos en las figuras 4.30 y 4.31 donde todas las líneas de comportamiento están casi superpuestas entre sí; por otro lado en la figura 4.32 se muestra la evolución de puntajes en general, donde se aprecia que el puntaje más bajo fue alcanzado por un algoritmo de enjambre de partículas y que la mayoría de las pruebas llegan a un resultado final que ronda los 24500 de puntaje.

Figura 4.30
Mejores pruebas escalón de 3.14 rad en general

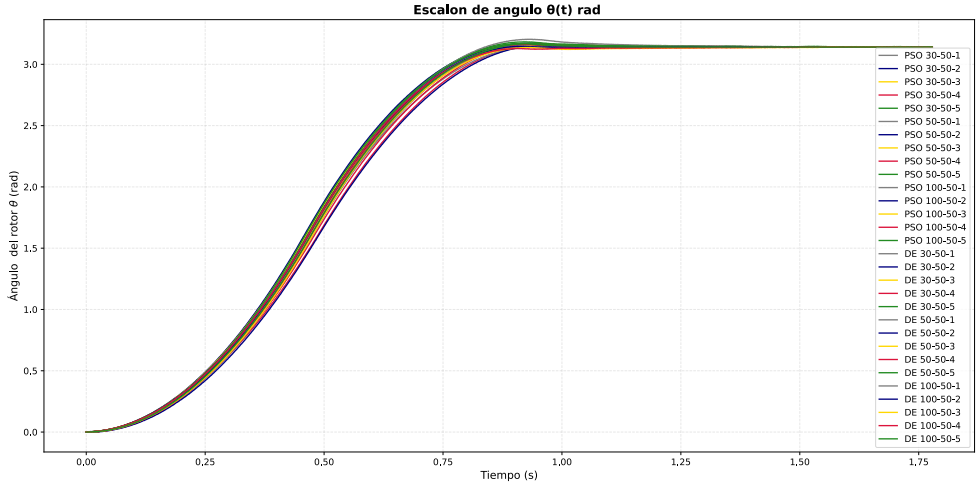


Figura 4.31
Mejores curvas de las iteraciones

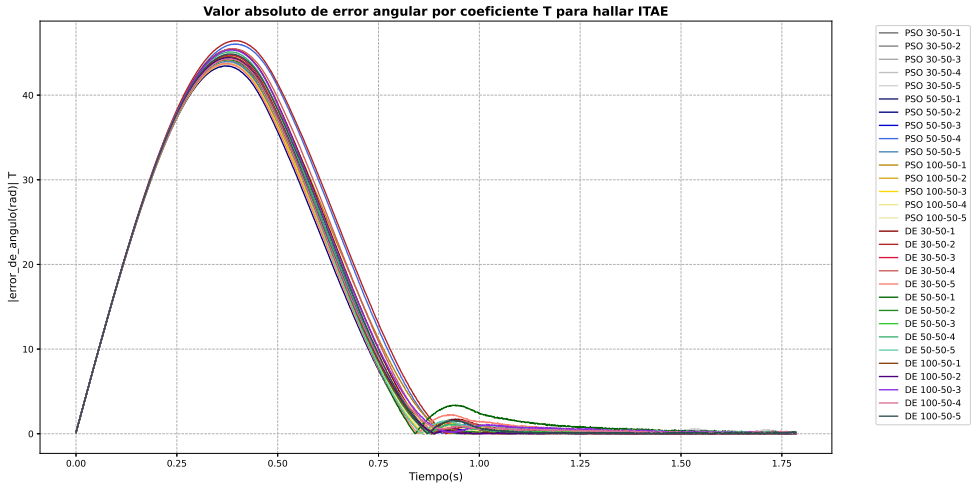
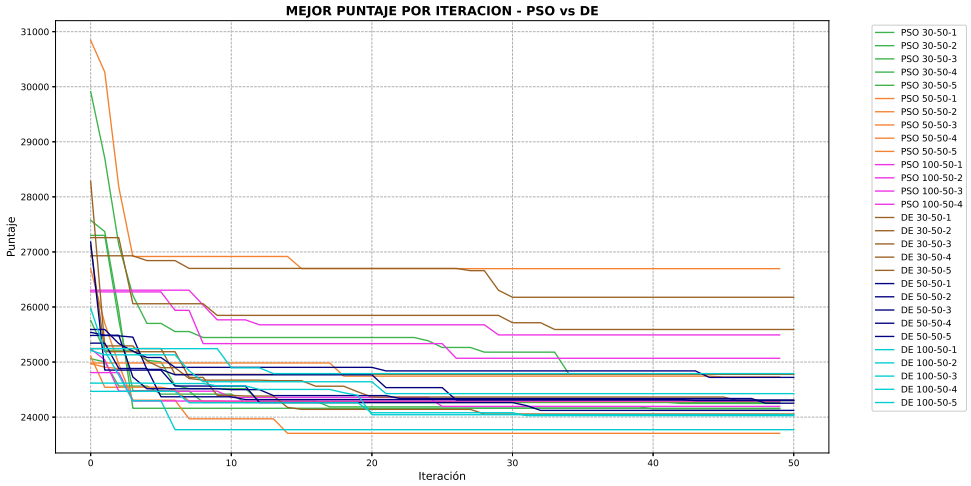


Figura 4.32
Mejores puntajes de cada prueba



4.4.4. Resultados de aplicación de los algoritmos de optimización

Las tablas de resultados fueron basadas en las características listadas, el puntaje final de cada prueba según la función ITAE y cual fue la última generación en la que el puntaje tuvo una mejora.

- Ts : Tiempo de establecimiento en segundos.
- ITAE: Menor valor de la función de optimización calculada con la ecuación 4.8.
- Mp: Máximo sobreimpulso en porcentaje del escalón total.
- EES: Error en estado estacionario en porcentaje del escalón total.
- Lg: Ultima generación con mejora en el puntaje.

En las tablas 4.1 y 4.2 se apreció un comportamiento muy variado en cuanto al sobreimpulso obtenido, el cual osciló entre 0.1 % y 2.04 %. Este último valor constituyó una excepción, puesto que ninguna otra prueba superó el 2 % en toda la investigación. Además, no se identificaron patrones resaltantes, salvo que, a mayor población inicial utilizada, menor fue el sobreimpulso final. Por otro lado, el error en estado estacionario presentó valores siempre menores o iguales a 0.07, los cuales resultaron insignificantes y fueron atribuidos principalmente al ruido de medición. Finalmente, respecto al tiempo de establecimiento, no se

observaron diferencias considerables; únicamente se destacó que, a mayor población inicial utilizada, se obtuvo un tiempo de establecimiento menor y más consistente entre las pruebas.

En las tablas 4.3 y 4.4 se apreció un comportamiento más consistente en cuanto al sobreimpulso del sistema al variar el número de partículas iniciales por iteración, ya que la tendencia no mostró una mejora al utilizar 100 partículas en comparación con la prueba realizada con 50. El error en estado estacionario también fue más consistente que en el caso de Evolución Diferencial, obteniéndose siempre valores menores o iguales a 0.05 %, los cuales resultaron producto del ruido de medición.

En la tabla de promedios 4.5 se apreció que, respecto a la característica de sobreimpulso, el valor promedio nunca superó el 0.9 %, por lo que este se mantuvo siempre dentro de la banda de tolerancia del 2 %, a partir de la cual se consideró finalizado el régimen transitorio. Asimismo, el error promedio en estado estacionario tampoco superó el 0.05 %, por lo que se consideró insignificante y sin presencia de oscilaciones pronunciadas. Por otro lado, el tiempo de establecimiento promedio se encontró entre 0.816 y 0.844 s, evidenciando un comportamiento similar entre las pruebas, con una variación del 3 %.

Tabla 4.1
Resultados de desempeño del algoritmo DE para 30 y 50

Resultados del algoritmo DE (30 y 50)										
Índice	30-1	30-2	30-3	30-4	30-5	50-1	50-2	50-3	50-4	50-5
Ts	0.829	0.859	0.821	0.836	0.804	0.947	0.818	0.825	0.823	0.809
ITAE	24775	26176	24301	25591	24056	24721	24311	24300	24252	24122
Mp	1.02	0.57	0.38	0.96	1.4	2.04	0.22	0.32	0.70	0.92
EES	0.02	0.03	0.03	0.04	0.04	0.07	0.03	0.04	0.05	0.04
Lg	6	30	46	33	28	44	26	43	48	32

Tabla 4.2
Resultados de desempeño del algoritmo DE para 100

Resultados del algoritmo DE (100)					
Índice	100-1	100-2	100-3	100-4	100-5
Ts	0.821	0.823	0.839	0.830	0.823
ITAE	23771	24043	24426	24026	24790
Mp	0.32	0.22	0.13	0.1	0.99
EES	0.02	0.02	0.05	0.04	0.03
Lg	6	20	21	31	13

Tabla 4.3
Resultados de desempeño del algoritmo PSO para 30 y 50

Resultados del algoritmo PSO (30 y 50)										
Índice	30-1	30-2	30-3	30-4	30-5	50-1	50-2	50-3	50-4	50-5
Ts	0.808	0.820	0.816	0.823	0.814	0.821	0.808	0.824	0.852	0.810
ITAE	24248	24787	24149	24266	24160	24283	23705	24728	26696	24281
Mp	0.89	1.05	0.41	0.25	0.61	0.29	0.61	0.45	0.02	0.83
EES	0.02	0.03	0.02	0.02	0.01	0.02	0.02	0.03	0.03	0.04
Lg	42	34	40	11	3	10	14	45	15	6

Tabla 4.4
Resultados de desempeño del algoritmo PSO para 100

Resultados del algoritmo PSO (100)					
Índice	100-1	100-2	100-3	100-4	100-5
Ts	0.849	0.810	0.812	0.831	0.814
ITAE	25492	24292	24194	25068	24413
Mp	0.22	0.76	0.80	0.64	0.67
EES	0.03	0.02	0.02	0.02	0.03
Lg	30	3	25	26	13

Tabla 4.5
Promedios de los índices de desempeño para los algoritmos DE y PSO

PROMEDIOS						
ÍNDICE	**DE**			**PSO**		
	30	**50**	**100**	**30**	**50**	**100**
TS	0.830	0.844	0.827	0.816	0.823	0.823
ITAE	24979.8	24341.2	24211.2	24322.0	24738.6	24691.8
MP	0.87	0.84	0.35	0.64	0.44	0.62
EES	0.03	0.05	0.03	0.04	0.03	0.02
Lg	28.6	38.6	18.2	26.0	18.0	19.0

4.5. RESULTADOS

4.5.1. Resultados para el objetivo de diseñar criterios de optimización para los coeficientes de los controladores basados en características del sistema.

Los criterios de optimización diseñados fueron los rangos o zonas de búsqueda dentro de las cuales se ejecutaron los algoritmos de optimización de la investigación. Para esto, se tomaron los coeficientes para los controladores basados en características físicas medidas del sistema, tales como:

$$R = 0.111\Omega, L = 115.6uH, J = 0.029Kgm^2, Kt = 0.0848\frac{Nm}{A} \tag{4.9}$$

De estos valores base se obtuvieron límites para los rangos de búsqueda de los coeficientes, asumiendo un sistema lineal e independiente de las incertidumbres de fricción, muestreo o límites físicos de corriente y velocidad en la sección 4.2. Estos valores fueron los que establecieron los criterios base para la región de optimización del sistema.

$$P_{posicion} = 0-5; P_{velocidad} = 0-34.1; I_{velocidad} = 0-828; \tag{4.10}$$

Además, de los valores de resistencia e inductancia se obtuvieron las constantes fijas para el lazo de corriente,las cuales mantienen el ancho de banda lo suficientemente alto para que no influya en el control de posición.

$$P_{corriente} = 0.115 \tag{4.11}$$

$$I_{corriente} = 111 \tag{4.12}$$

4.5.2. Resultados respecto al objetivo de implementar una planta que emule un servosistema para el control de un motor sin escobillas de corriente continua (BLDC), determinando las características y especificaciones de los componentes adecuados que la componen.

La identificación se realizó para el tipo de controlador utilizado y el modelo matemático del motor usado en la investigación. Este modelo fue el siguiente.

$$v_d = R_s i_d + L_d \frac{di_d}{dt} - P\omega_m i_q L_q \tag{4.13}$$

$$v_q = R_s i_q + L_q \frac{di_q}{dt} + P\omega_m (i_d L_d + \lambda_m) \tag{4.14}$$

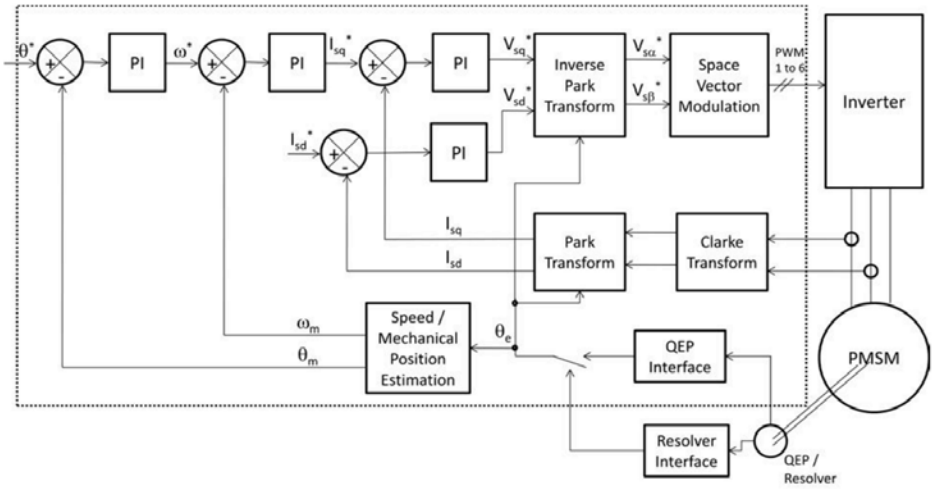
$$v_0 = R_s i_0 + L_0 \frac{di_0}{dt} \tag{4.15}$$

$$T = \frac{3}{2} N \left(i_q (i_d L_d + \lambda_m) - i_d i_q L_q \right) \tag{4.16}$$

Además, se utilizó el tipo de controlador de campo orientado (FOC), para el cual el diagrama base se encuentra en la figura 4.33.

Figura 4.33

Diagrama de control de posición de BLDC



Nota. *Adaptado de Texas Instruments (2021)*

El resultado de la identificación de componentes resultó en las siguientes características de implementación en la placa de control y motor utilizados para realizar las pruebas; según el modelo matemático y tipo de control se tienen como características indispensables.

- El sistema debe poseer una forma de medición de la posición del motor, la cual se puede realizar con algoritmos o un encoder; se utilizó un encoder para la presente investigación.
- El sistema debe ser capaz de realizar un monitoreo de corriente de al menos dos de las tres fases del motor; estos valores se usaron para ejecutar la realimentación necesaria en el lazo de corriente del control FOC.
- Se controló un motor con onda contraelectromotriz sinusoidal, ya que si fuese trapezoidal, el modelo matemático se alteraría y el uso del control FOC se vería afectado.

Por las características mencionadas se escogieron los siguientes componentes que cumplen con dichas condiciones.

- Motor Eaglepower LA8308, que posee onda contraelectromotriz sinusoidal y es un motor sin escobillas.
- Placa MKS XRIVE MINI, que posee un encoder magnético integrado para la medición del ángulo absoluto, además de medidores de corriente integrados.

Con estas características obligatorias se puede realizar el control del sistema; el resto de características, si bien convenientes, no son indispensables para realizar pruebas para motores sin escobillas mediante un controlador de campo orientado. Estas otras características son:

- Manejo de corriente de la placa superior al límite máximo de corriente soportado por el motor para aprovechar al máximo el torque generado; de no ser cumplida esta característica limita el torque que puede ser usado. Sin embargo, no limita la posibilidad de realizar el control del movimiento del motor.

- Microcontrolador cuyo desempeño exceda el ancho de banda en los lazos de corriente y velocidad. Esta característica ayuda a evitar los problemas de muestreo asociados al control de alta velocidad de corriente. No obstante, si no se tiene un funcionamiento a alta velocidad de rotación, se pueden mitigar los efectos disminuyendo el ancho de banda de corriente para el sistema en general.

4.5.3. Resultados para el objetivo de Validar las técnicas de optimización seleccionadas y comprobar cuál es más efectiva mediante resultados que permitan su comparación.

Las técnicas de optimización se validaron mediante pruebas de desempeño; antes de esto, se realizaron pruebas previas en las cuales se seleccionaron los coeficientes para los algoritmos PSO y DE que permitieran la mayor exploración posible y una convergencia al mejor valor general a lo largo de las iteraciones, tal como se mostró en la sección 5.2.1. De las cuales se obtuvieron los valores para los coeficientes de los algoritmos de:

$$W = 0.5, \quad C1 = 2, \quad C2 = 0.5 \qquad \text{(Parámetros PSO)}$$

$$F = 0.2, \quad CW = 0.5 \qquad \text{(Parámetros DE)}$$

Con estos valores se realizaron pruebas donde lo que se cambió fue el número de partículas iniciales para cada algoritmo. Estos números fueron 30, 50 y 100, los cuales se probaron en 50 generaciones. Esto resultó en los datos comparables de las figuras 4.31 y 4.32, además de la tabla 4.5.

Según los gráficos y datos antes mencionados, se obtuvieron los siguientes resultados.

- Se validó el funcionamiento de ambos algoritmos, puesto que los mismos siempre lograron la disminución del puntaje evaluado.
- El algoritmo que logró el mejor puntaje general es el de enjambre de partículas con 50

partículas iniciales y un puntaje de 23705.

- El algoritmo que logró mejor puntaje promedio final fue el de evolución diferencial con un promedio de 24510, que superó por 0.3 % el promedio del algoritmo de enjambre de partículas.
- El algoritmo que menos iteraciones empleó en general para llegar a sus mejores puntajes en general fue el de enjambre de partículas con 21 iteraciones en promedio, disminuyendo el promedio del otro algoritmo en un 33 %.

4.5.4. Resultados para el objetivo general de diseñar e implementar un controlador de posición para motores sin escobillas de corriente continua, el cual optimice los parámetros del controlador mediante algoritmos de PSO y algoritmos de evolución diferencial

Este objetivo se logró mediante la implementación de la planta con los componentes adecuados y la realización de pruebas de optimización para un escalón de 3.14 rad (180° sexagesimales). Los resultados presentados en las figuras 4.30 y 4.32 demostraron que las técnicas de optimización aplicadas funcionaron correctamente, ya que, en general, evidenciaron un comportamiento adecuado tanto en el régimen transitorio como en el estado estacionario del sistema de control de posición del motor. Esto permitió comprobar que el diseño de los límites de búsqueda fue efectivo, evitando características indeseables y alcanzando un desempeño consistente en los distintos algoritmos evaluados.

Las principales características observadas fueron las siguientes:

- Error de estado estacionario mínimo.
- Sobreimpulso inferior al 2 % en prácticamente todas las pruebas.
- Respuesta estable durante el régimen transitorio y el estado estacionario.
- Ausencia de oscilaciones o vibraciones apreciables durante la respuesta.

- Seguimiento adecuado de la referencia de 3.14 rad.
- Comportamiento repetitivo y consistente entre las diferentes optimizaciones.

Por otro lado, y específicamente para los dos algoritmos probados, se obtuvieron las diferencias más resaltantes:

- El algoritmo que obtuvo, en general, los mejores puntajes de la función de evaluación fue el de evolución diferencial, con un promedio de 24510, superando al otro algoritmo en un 0.3 %.
- El algoritmo que requirió un menor número de iteraciones para alcanzar su valor más bajo fue el de enjambre de partículas, con 21 iteraciones, disminuyendo en un 33 % el promedio de iteraciones del otro algoritmo.

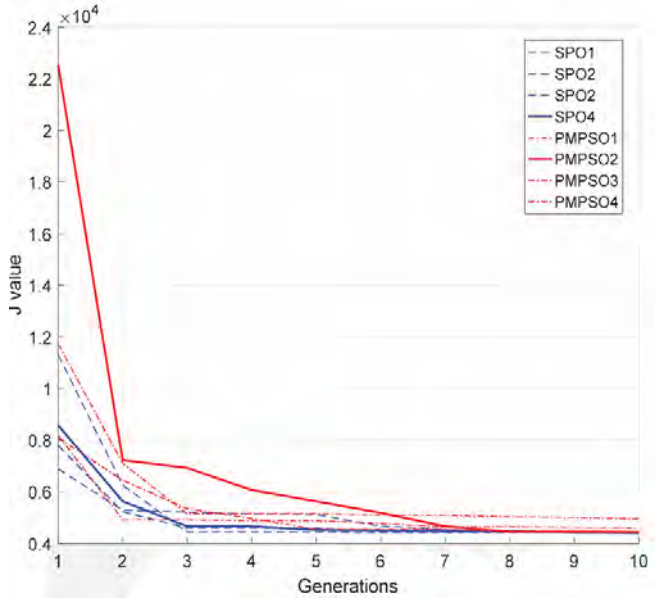
Por lo tanto, de esta comparación se concluyó que el algoritmo más efectivo para la optimización de la planta fue el de enjambre de partículas, debido a que presentó una ventaja más notable al requerir un menor número de iteraciones para alcanzar su mejor puntaje.

4.6. Discusión

4.6.1. Descripción de los hallazgos más relevantes y significativos

En primer lugar se halló que las optimizaciones, a diferencia de otras investigaciones donde se prueban algoritmos de optimización en simulaciones, no tienen un punto determinado mínimo, ya que el sistema real presenta varios puntos mínimos a veces separados por distancias considerables y significativas, esto se mostró en que las soluciones de las optimizaciones tienden a diferentes puntajes como se muestra en la figura 4.32 a diferencia de una simulación donde se suelen tender exactamente a un mismo puntaje mínimo. Esto se muestra en la investigación Tien et al. (2021) donde en 6 de las 8 pruebas que se realizaron coinciden a un mismo puntaje mínimo como se muestra en la figura 4.34.

Figura 4.34
Evolución de puntaje de optimización



Nota. *Adaptado de Tien et al. (2021)*

4.6.2. Comparación crítica con la literatura existente

Se comparó el resultado de manera cuantitativa con la literatura existente, respecto al trabajo de Akash (2025), donde se mostró el control de un servomotor DC mediante técnicas de realimentación de estados (SFC) y por realimentación de estados con acción integral (SFCIA). Esta investigación se enfocó en la optimización del control de posición, además de usar una prueba escalón para la comprobación de su efectividad, obteniendo como resultados.

- Error en estado estacionario de 0 % y máximo sobreimpulso menor a 2 % para el caso de la realimentación de estados con acción integral como se muestra en la subfigura (b) de la figura 4.35.
- Error en estado estacionario de 0 % y máximo sobreimpulso de 2.39 % para el control de realimentación de estados como se muestra en la subfigura (a) de la figura 4.35.

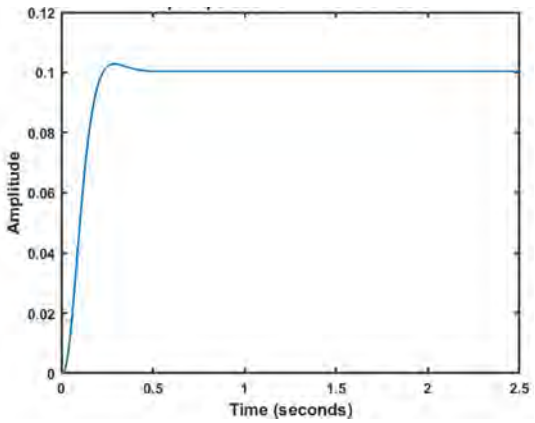
Esto muestra que los resultados de la presente investigación superaron al método de control de realimentación de estados y se encontraron a la par del método de realimentación de estados con acción integral, debido a que el sobreimpulso promedio obtenido de las pruebas no superó el 1 % y el error en estado estacionario presentó valores en promedio entre el

0.03 % y 0.05 %, estos resultados son propios de una prueba real, en la que siempre existe un cierto nivel de ruido a diferencia de una prueba ideal donde se puede obtener un error ideal de 0 para el estado estacionario.

Figura 4.35

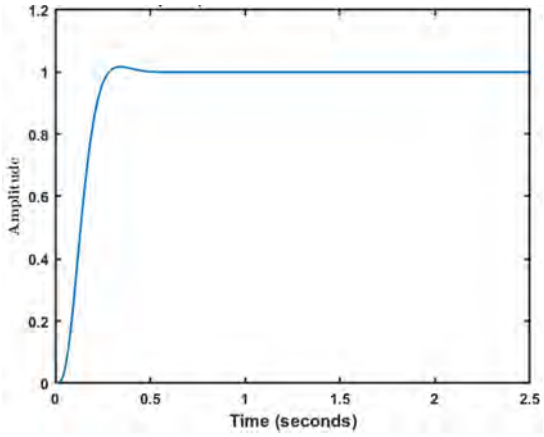
Gráficas de pruebas con los métodos comparados

(a) *Prueba escalon para control SFC*



Nota. *Adaptado de Akash (2025)*

(b) *Prueba escalon para control SFCIA*



Nota. *Adaptado de Akash (2025)*

Por otro lado se realizó una comparación cuantitativa con el trabajo de Tien et al. (2021), donde se realizó la optimización para el control de posición de un controlador FOC mediante técnicas de enjambre de partículas(PSO) y optimización por enjambre de partículas con múltiples poblaciones en paralelo(PM-PSO), aplicadas a pruebas en una simulación en MATLAB de donde se obtuvo la figura 4.36 y los siguientes resultados:

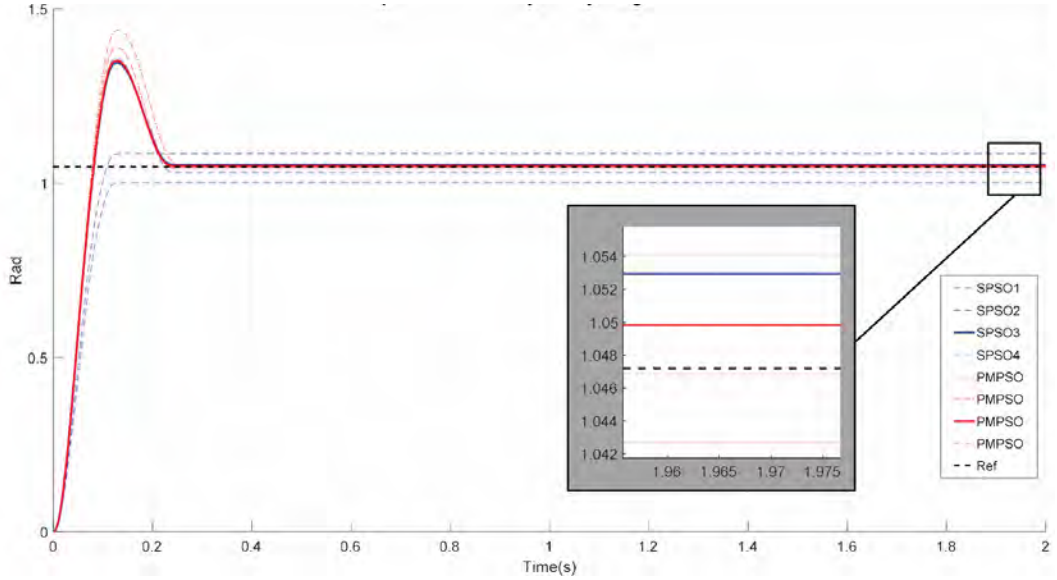
- Error en estado estacionario de 0.57 % y máximo sobreimpulso de 23.85 % para el caso del algoritmo de enjambre de partículas.
- Error en estado estacionario de 0.19 % y máximo sobreimpulso de 23.74 % para el caso del algoritmo de enjambre de partículas con múltiples poblaciones.

Los resultados mostraron que la investigación actual obtuvo resultados mejores a ambos métodos probados, esto se atribuyó principalmente a que la región de búsqueda usada en el trabajo mencionado fue una región arbitraria, no partió de un análisis previo de posibles soluciones como en la investigación actual, además de que la optimización general del trabajo

citado se realizó incluyendo la optimización simultánea del lazo de corriente, lo cual provocó un universo de búsqueda mucho más grande y variable, cosa que se evitó en la presente investigación al optimizar solo los lazos de velocidad y posición aislando el de corriente.

Figura 4.36

Pruebas de escalón de la investigación citada



Nota. *Adaptado de Tien et al. (2021)*

CONCLUSIONES

- Se concluyó de las pruebas realizadas, en general, que la planta fue funcional y permitió la comparación mediante pruebas de optimización con respuesta al escalón. El algoritmo de evolución diferencial obtuvo el mejor puntaje promedio de la función ITAE, con 24510 y una mejora del 0.3 % respecto al otro. El algoritmo que utilizó menos iteraciones, en promedio, fue el de enjambre de partículas, con 21 iteraciones y una mejora del 33.3 % respecto al otro. Respecto al sobreimpulso (Ms), el error de estado estacionario (EES) y el tiempo de establecimiento (Ts), sus valores resultaron bastante similares, sin diferencias significativas entre sí. Por lo tanto, debido a que la diferencia en el puntaje fue mucho menor que la ventaja en el número de iteraciones, se concluyó que el algoritmo más efectivo para la optimización de la planta fue el de enjambre de partículas.

- Se plantearon criterios válidos para la selección de rangos de identificación necesarios para los algoritmos de evolución diferencial y de enjambre de partículas, delimitando los coeficientes proporcional e integral que rigieron los controladores de los lazos de velocidad y posición. Para ello, se utilizó un modelo aproximado basado en las características físicas del mismo: $Rs = 0.111\Omega; Ls = 115.6uH; J = 0.029kgm^2; Kt = 0.0848\frac{Nm}{A}$. Estos límites fueron $P_{velocidad} = 0 - 34.1; I_{velocidad} = 0 - 828; P_{posicion} = 0 - 5$, multiplicados por un valor arbitrario de 1.5. Estos límites demostraron representar un universo de soluciones válido, ya que los parámetros hallados nunca tendieron a salir de los límites planteados.

- Se realizó la correcta implementación mediante la selección de características esenciales y complementarias que permitieron el correcto funcionamiento de los componentes y su respectiva optimización mediante los algoritmos, tales como el motor Eaglepower LA8308 KV90 y el controlador de potencia ODRIVE MINI, encargado del control del voltaje de entrada. Siendo las características fundamentales que el motor estuviera diseñado para trabajos con alto torque, como los motores de VANT agrícolas, y que tuviera una onda contraelectromotriz sinusoidal para mantener coherencia con el mo-

delo matemático; por otra parte, la placa de control y potencia debía poder monitorizar los valores de corriente y posición para que el control de campo orientado (FOC) fuera posible.

- El sistema implementado en el presente estudio cumplió con la optimización del control de posición de un motor sin escobillas, esto evidenciado en las pruebas, donde los algoritmos siempre redujeron el puntaje inicial de la función, logrando errores de estado estacionario menores al 0.05 % y un sobreimpulso menor al 2 %, exceptuando una prueba de Evolución Diferencial, donde se obtuvo un 2.04 %, constituyendo una excepción y no la regla; con lo cual siempre se llegó de manera precisa al ángulo objetivo. Estos valores se lograron de manera consistente luego de las optimizaciones con los algoritmos de enjambre de partículas y evolución diferencial, lo cual demostró su efectividad para optimizar los parámetros del controlador.

RECOMENDACIONES

- Para futuros trabajos se recomienda el uso de técnicas de control como el Sliding Mode Control (SMC), aplicado a los controladores de cada lazo del FOC, lo cual permitirá controlar cargas no lineales y condiciones variables del sistema. Además, se recomienda adaptar la técnica de PSO, ya que en el presente trabajo demostró un funcionamiento correcto en la optimización de coeficientes.

- Se recomienda la investigación de otras formas de búsqueda de rangos de coeficientes, ya sea agregando complejidad al modelo matemático utilizado mediante la introducción de la fricción o asociando los coeficientes de control más relacionados, como los de los lazos de velocidad y posición, logrando un menor número de coeficientes. Esto permitiría simplificar y acelerar la optimización en general, la cual, en el presente trabajo, tuvo una duración del orden de 45 minutos a 3,5 horas.

- Se recomienda implementar métodos para reducir los ruidos de conmutación de los transistores, ya sea mediante hardware o software. Asimismo, por parte del hardware se recomienda utilizar controladores de potencia cuyas características excedan los requerimientos del motor empleado, puesto que, si no cuentan con un amplio margen de seguridad, se pueden presentar fallas que provoquen la destrucción de los componentes electrónicos.

BIBLIOGRAFÍA

Abdelgar, M., Ramadan, H., Khalil, A., H. Farag, M. Bahgat, O. R., and El-Shaer, Y. (2023). Optimization of pi-cascaded controller's parameters for linear servo mechanism: A comparative study of multiple algorithms. *IEEE*.

Adela, J. (2023). *CONTROL VECTORIAL DE MOTOR BLDC PARA SIMULADOR DE INSPECCION DE AGUJAS HIPODÉRMICAS*. Tesis de licenciatura, UNIVERSIDAD POLITECNICA DE MADRID, Madrid, España.

Ahmad, M. F., Isa, N. A. M., Lim, W. H., and Ang, K. M. (2021). Differential evolution:a recent review based on state of the art works. *Alexandria Engineering Journal*.

Akash, R. K. (2025). Comparative analysis of control strategies for position regulation in dc servo motors. *arxiv*.

Chavan, P. L., Gowda, D., and Nayak, S. K. (2023). Field oriented control technique for pmsm. *Atlantis Press*.

Cube Mars (n.d.). Quasi direct drive motor higher torque, designed for robotic joints. `https://www.cubemars.com/categorys/quasi-direct-drive-motor`. Accessed: 2025-04-01.

Djouadi, H., Ouari, K., Belkhier, Y., Lehouche, H., Bajaj, M., and Blazek, V. (2024). Improved robust model predictive control for pmsm using backstepping control and incorporating integral action with experimental validation. *Department of Electrical Engineering, Graphic Era*.

Dorf, R. C. (2011). *MODERN CONTROL SYSTEMS*. PEARSON, United States of America.

Eaglepower (n.d.). Eaglepower la 8308 kv90 motor. `https://robu.in/product/eaglepower-la-8308-kv90-motor/`. Robu.in product page.

Guong, R. L. L., Hisham, S. B., Elamvazuthi, I., Aole, S., Parasuraman, S., and Khan, M. A. (2018). Pid tuning of process plant using particle swarm optimizationl. *IEEE*.

Infineon Technologies (2023). *AN2023-10: Basic Motor Parameters and the Configuration*. Infineon Technologies AG, 1.0 edition.

Jain, M., Saihjpal, V., Singh, N., Singh, S. B., Singh, S. B., and Singh, S. B. (2022). An overview of variants and advancements of pso algorithm. *applied sciences*.

Jinlong, Y., Baochao, W., Dongwei, W., Ning, J., Xueguan, Z., and Fengmei, L. (2024). Design and testing of a low-cost mobile platform for contactless building surveying. *Journal of Engineering Research*.

Kim, H. and Asbeck, A. T. (2020). An elbow exoskeleton for haptic feedback made with a direct drive hobby motor. *HardwareX*.

Lajić, R. and Matić, P. (2023). Digital position control system with a bldc motor using field oriented control. *INFOTEH-JAHORINA*.

Landolfi, E. and Natale, C. (2023). An adaptive cascade predictive control strategy forconnected and automated vehicles. *WILEY*.

Luo, R., Wang, Z., and Sun, Y. (2022). Optimized luenberger observer-based pmsm sensorless control by pso. *Hindawi*.

Makerbase (2023). Makerbase xdrive mini high-precision brushless servo motor controller, based on odrive3.6 with as5047p on board. Diagramas y repositiorio de la placa de Makerbase. Accessed: 2025-04-01.

Makerbase (2024). Productos compatibles con esp32. Documentación oficial del proyecto SimpleFOC. Accedido: 2025-04-01.

Mandra, S. (2014). Comparison of automatically tuned cascade control systems of servodrives for numerically controlled machine tools. *ELEKTRONIKA IR ELEKTROTECHNIKA*.

Matevosyan, R. (2021). Control vectorial del par motor de un motor brushless. Trabajo de titulación, UNIVERSIDAD POLITÉCNICA DE VALENCIA, Valencia, España.

MATLAB (2025a). Modulación de vector espacial (svm) para sistemas de control de motores.

MATLAB (2025b). Pmsm. Versión R2024b.

Microchip Technology Inc. (2023). *Motor Control Application Framework*.

Mohanraj, D., Aruldavid, R., Belkhier, Y., Lehouche, H., Bajaj, M., and Blazek, V. (2022). A review of bldc motor: State of art, advanced control techniques, and applications. *IEEE*.

Mukhtar, A., Tayal, V. K., and Singh, H. (2019). Pso optimized pid controller design for the process liquid level control. *RDCAPE*.

Mustafa, N. and Hashim, F. H. (2020). Design of a predictive pid controller using particle swarm optimization. *INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS*.

ODrive Robotics, I. (2025). Legal notices - odrive robotics documentation. Accessed: 2025-06-05.

Ogata, K. (2010). *Ingeniería de control moderna*. Pearson, Madrid.

Park, H.-J., Ahn, H.-W., and Go, S.-C. (2023). A study on performance and characteristic analysis according to the operating point of ipmsm drive. *Energies*.

Parque, V. and Khalifa, A. (2023). Pid tuning using differential evolution with success-based particle adaptations. *IEEE*.

Robotics, O. (n.d.). Motor with encoder magnet - m8325s 100kv. `https://shop.odriverobotics.com/collections/motors/products/m8325s`. Product page from ODrive Robotics. Accessed: 2025-04-01.

Ruppert, F. and Badri-Spröwitz, A. (2022). Learning plastic matching of robot dynamics in closed-loop central pattern generators. *nature machine intelligence*.

Sahdev, S. (2018). *ELECTRICAL MACHINES*. Cambridge, United Kingdom.

SimpleFOC Project (2024). *Arduino SimpleFOCShield v3.2: An open-source low-cost Brushless DC motor driver board*. Documentación oficial del proyecto SimpleFOC. Accedido: 2025-04-01.

Skuric, A., Bank, H. S., Unger, R., Williams, O., and Reyes, D. G. (2022). Simplefoc: A field oriented control (foc) library for controlling brushless direct current (bldc) and stepper motors. *JOOS*.

Slowik, A. and Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. *Springer*.

STMicroelectronics (2018). Motor control part1 - 2 how to measure motor parameters. Recuperado de YouTube.

STMicroelectronics (2021). *Advanced BLDC controller with embedded STM32 MCU evaluation board*. STMicroelectronics.

Storn, R. and Price, K. (1997). Differential evolution– a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*.

Tawfiq, K. B., Sergeant, P., and Mansour, A. S. (2024). Comparative analysis of space vector pulse-width modulation techniques of three-phase inverter to minimize common mode voltage and/or switching losses. *Mathematics*.

Texas Instruments (2021). Instaspin-foc and instaspin-motion. *Texas Instruments*.

Tien, N., Van, C., and Huy, H. (2021). Optimal foc-pid parameters of bldc motor system control using parallel pm-pso optimization technique. *ATLANTIS PRESS*.

Wang, Z., Jawahar, S., Riccobono, A., and Turevskiy, A. (2022). Cascade digital pid control design for power electronic converters. *MathWorks*.

Wilson, D. (2023). Teaching your pi controller to behave (part ii). *Texas Instruments*.

Zhao, Y., Qiu, Q., and Zhao, X. (2021). Pid parameter tuning for buck controllers based on an improved differential evolution algorithm. *IOP Publishing*.

Zhou, Y., Yongming, B., Yifan, Z., and Li, C. (2025). An intelligent optimization algorithm with novel fitness function for high-performance pmsm foc. *Alexandria Engineering Journal*.

ANEXOS

Anexo 1. Medidas de ángulos base y de encoder

Tabla 4.6
Comparativa entre el valor patrón y el valor medido por el encoder

Valor patrón	Valor encoder
0	0.0
10	10.3
20	20.1
30	29.8
40	40.1
50	49.8
60	60.2
70	69.9
80	80.2
90	90.0
100	100.3
110	110.6
120	120.9
130	131.2
140	140.9
150	151.3
160	161.6
170	171.3
180	181.6
190	191.4
200	201.1
210	210.8
220	220.6
230	230.3
240	240.1
250	250.4
260	260.1
270	269.9
280	280.2
290	289.9
300	299.7
310	310.0
320	319.7
330	329.6
340	339.9
350	350.2

Anexo 2. CODIGO ARDUINO PARA EL MICROCONTROLADOR STM405

```

#include <SimpleFOC.h>

// Odrive M0 pines de placa
#define M0_INH_A PA8
#define M0_INH_B PA9
#define M0_INH_C PA10
#define M0_INL_A PB13
#define M0_INL_B PB14
#define M0_INL_C PB15
// M0 pines sensores de corriente
#define M0_IB PC0
#define M0_IC PC1
// Odrive M0 pines de encoder
#define M0_ENC_A PB4
#define M0_ENC_B PB5
#define M0_ENC_Z PC9

// Pin de habilitacion
#define EN_GATE PB12

// SPI pines de comunicacion SPI
#define SPI3_SCL  PC10
#define SPI3_MISO PC11
#define SPI3_MOSO PC12


// Instancia de motor
BLDCMotor motor = BLDCMotor(20,0.111,115.7);
BLDCDriver6PWM driver = BLDCDriver6PWM(M0_INH_A,M0_INL_A, M0_INH_B,
    M0_INL_B, M0_INH_C,M0_INL_C, EN_GATE);
float target_angle = 0;
int ciclo = 0;
// Coeficientes Extras agregados
long tiempo=0;
float nueva_posicion = 0.0;
float datos[5];  // Arreglo para almacenar 5 valores
bool posicion_actualizada = false;
bool datos_recibidos = false;

String entrada = "";
//////////////////////////////////////////////////
// 0.0005 Ohm resistencia shunt
// Ganancia de  10
// Lectura de corriente de fase b y c, no se conecta la fase A
LowsideCurrentSense current_sense = LowsideCurrentSense(0.0005f, 10.0
    f, _NC, M0_IB, M0_IC);

// MagneticSensorSPI(int cs, float _cpr, int _angle_register)
// config            - SPI config
//  cs               - SPI chip select pin
//MagneticSensorSPI sensor = MagneticSensorSPI(AS5147_SPI, M0_ENC_A);
MagneticSensorSPI sensor = MagneticSensorSPI(AS5147_SPI, PA15);
SPIClass SPI_3(SPI3_MOSO, SPI3_MISO, SPI3_SCL);

void setup() {
```

```
  ////////////////////////////////////////////////////
  motor.torque_controller = TorqueControlType::foc_current;

  motor.PID_current_q.P = 0.116f;//Coeficientes para corriente
  motor.PID_current_q.I= 111;
  motor.PID_current_d.P= 0.116f;
  motor.PID_current_d.I = 111;
  motor.LPF_current_q.Tf = 0.0004f;
  motor.LPF_current_d.Tf = 0.0004f;
  motor.current_limit = 5;
  ////////////////////////////////////////////////////
  // Definiendo los baudios para monitorizar
  Serial.begin(1000000);
  // Frecuencia de pwm a usar [Hz]
  driver.pwm_frequency = 20000;
  // Voltaje de alimentacion [V]
  driver.voltage_power_supply = 12;
  // Max DC voltage permitido
  driver.voltage_limit = 3.0;
  // Inicializacion de driver
  driver.init();
  // link the motor y el driver
  motor.linkDriver(&driver);
  // Iniciar magnetic sensor hardware
  sensor.init(&SPI_3);
  // link  motor a sensor
  motor.linkSensor(&sensor);
  // Sellecciona el tipo de ciclo y control de torque
  motor.foc_modulation = FOCModulationType::SpaceVectorPWM;
  //Tipo de control de motor, ANGULO DE ROTOR
  motor.controller = MotionControlType::angle;
  //De acuerdo al motor modifica los coeficientes del controlador
  motor.PID_velocity.P = 28.9;
  motor.PID_velocity.I = 200.6;
  //P del angulo de rotor
  motor.P_angle.P = 1;
  motor.LPF_velocity.Tf = 0.001f;
  motor.velocity_limit = 60;
  // Maximo voltaje permitido
  motor.voltage_limit = 3.0;
  // Limite de voltaje para proceso de alineacion inicial
  motor.voltage_sensor_align = 1;
  // Inicio de motor
  motor.init();

  // Realizando conexion al driver
  current_sense.linkDriver(&driver);
  // Iniciando el sensado de corriente
  current_sense.init();
  current_sense.skip_align = true;
  motor.linkCurrentSense(&current_sense);

  // Alineando sensor e iniciando control FOC
  motor.initFOC();

  Serial.println(F("Motor ready."));
  Serial.println(F("Set the target angle using serial terminal:"));
  delay(1000);
}

```

```
115
116 void loop() {
117   if (ciclo < 20500) {//Devuelve 2050 datos por seguridad
118     if (ciclo % 10 == 0) {
119       if(motor.shaft_velocity>15){
120         while(1){
121           Serial.println("error fatal");
122           motor.disable();
123         }
124       }
125       Serial.println(motor.shaft_angle, 6);//transmicion de angulo
126       ciclo++;
127     }
128     else{
129       if(ciclo <=20500)//limitacion de transmicion
130       {
131         ciclo++;
132       }
133     }
134   }
135   motor.loopFOC();
136   motor.move(target_angle);
137
138   if (Serial.available()) {
139     String entrada = Serial.readStringUntil('\n');  // leer linea
          completa
140     entrada.trim(); // eliminar espacios o saltos de linea extra
141     if (entrada.indexOf(';') == -1) {
142       if (entrada.equalsIgnoreCase("MARCA")) {
143         Serial.print(motor.shaft_angle, 6);
144         Serial.print(";");
145         Serial.println(motor.shaft_velocity, 6);
146         return;
147       }
148       //  Caso: un solo nro
149       float numero = entrada.toFloat();
150       target_angle = numero;   // Proceso con un solo valor
151       ciclo=0;
152
153     } else {
154       // Caso: multiples valores separados por ';'
155       int pos1 = entrada.indexOf(';');
156       String dato1 = entrada.substring(0, pos1);          00 // busca
            primer valor
157       int pos2 = entrada.indexOf(';', pos1 + 1);
158       String dato2 = entrada.substring(pos1 + 1, pos2);    // busca
            segundo valor
159       String dato3 = entrada.substring(pos2 + 1);          // busca
            tercer valor
160
161       motor.P_angle.P=dato1.toFloat();
162       motor.PID_velocity.P=dato2.toFloat();
163       motor.PID_velocity.I=dato3.toFloat();
164
165     }
166   }
167 }
```

Anexo 3. CODIGO PYTHON PARA RECEPCION DE DATOS, EJECUCION DE PSO Y DE

Listing 4.1 *Funcion base de PSO*

```
1
2  ## Optimizacion de control de posicion con algoritmo de PSO
3  import serial
4  import time
5  import numpy as np
6  import matplotlib.pyplot as plt
7  import os
8  class SerialDE: #INICIO DE LA CLASE CON PARAMETROS TENPORALES
9    def __init__(self, port="COM10", baudrate=1000000, cantidad_recibida
       =100,
10     num_particulas=5, dimensions=2, bounds=(0, 1), max_iters=3,W=0.0,
         C1=0.0,C2=0.0):
11     # CONFIGURACION PSO, ASIGNACION DE PARAMETROS
12     self.W=W
13     self.C1=C1
14     self.C2=C2
15     # Configuracion de Serial, BAUDIOS , PORT, ETC.
16     self.port = port
17     self.baudrate = baudrate
18     self.cantidad_recibida = cantidad_recibida #INDICADOR DE CANTIDAD
         DE MUESTRAS
19     self.valor_referencia = 3.0  #VALOR DE REFERENCIA USADO PARA
         HALLAR EL ERROR DE POSICION
20     self.posicion_actual = 0.0  #VALOR DE ANGULO DE POSICION INICIAL
         EN RADIANES
21     self.mejor_curve_por_iter = [] #LISTA PARA GUARDAR MEJORES CURVAS
         HALLADAS
22     try:
23       # CONEXION INICIAL DE SERIAL Y ERROR DE CONEXION SI NO
           DISPONIBLE
24       self.ser = serial.Serial(self.port, self.baudrate, timeout=1)
25       print(f"[INFO] Serial abierto en {self.port} @ {self.baudrate}")
26     except Exception as e:
27       raise RuntimeError(f"No se pudo abrir puerto serial: {e}")
28
29     # CONFIGURACION DE PARAMETROS PSO
30     self.num_particulas = num_particulas #TOTAL DE PARTICULAS PARA
         ITERACIONES
31     self.dimensions = dimensions # DIMENSIONES TOTALES O NUMERO DE
         PARAMETROS (3)
32     self.bounds = bounds  # (min, max) LIMITES DE VALORES PARA LOS
         PARAMETROS NORMALIZADOS
33     self.max_iters = max_iters
34     # INICIALIZACION DE PARTICULAS, VALORES ALEATORIOS DE PARTICULAS
35     self.posiciones = np.random.uniform(bounds[0], bounds[1], (
         num_particulas, dimensions))
36     # AJUSTANDO LOS VALORES DE LAS PARTICULAS A LOS LIMITES
37     self.posiciones = np.clip(self.posiciones, bounds[0], bounds[1])
38     # INICIANDO VALORES DE VELOCIDADES DE PARTICULAS
39     self.velocities = np.zeros((num_particulas, dimensions))
40     # ASIGNACION DE MEJORES VALORES INDIVIDUALES
41     self.personal_best = self.posiciones.copy()
42     # PUNTAJE MUY ALTO INICIAL
43     self.personal_best_puntajes = np.full(num_particulas, np.inf)
44     # DECLARACION PERO NO DEFINICION DE LA MEJOR PARTICULA EN GENERAL
45     self.global_best = None
```

```
46      self.global_mejor_puntaje = np.inf  #PUNTAJE MUY ALTO DE MEJOR
            PARTICULA INICIAL
47
48      # HISTORIAL DE DATOS PARA GRAFICAR LA EVOLUCION DE LA OPTIMIZACION
49      self.mejor_puntaje_historico = [] # GUARDANDO PUNTAJES Y
            POSICIONES PARA GRAFICAS FINALES
50      self.posiciones_historico = []
51      self.coefcientes_string =""
52    #FUNCION DE CAPTURA DE DATOS PARA CADA PARTICULA PROBADA
53    def captura_de_datos(self, cmd="2"):
54      self.ser.reset_input_buffer() #LIMPIANDO ENTRADA DE SERIAL DE
            DATOS EXTRAS NO DESEADOS
55      ########################################################
56      self.funcion_posicion_actual() # FUNCION QUE ACTUALIZA LA POSICION
            ACTUAL REAL FISICA
57      self.ser.write((self.coefcientes_string + "\n").encode()) #
            ACTUALIZANDO PARAMETROS KPP,KIV,PKV
58      time.sleep(0.001)
59      self.ser.write((str(self.posicion_actual) + "\n").encode())#
            ENVIANDO NUEVA POCICION A MCU
60      print(self.posicion_actual)
61      print(f"[TX] {cmd} -> esperando {self.cantidad_recibida} muestras"
            ) #ESPERA DE MUESTRAS
62      datos = [] #DATOS ALMACENADOS PARA UNA SOLA PARTICULA
63      tiempo_inicio = time.time() #TIEMPO PARA EVITAR ERRORES DE
            EXCESIVO TIEMPO DE RECEPCION
64      espera_maxima = 2.5  # ESPERA EN SEGUNDOS PARA EVITAR ERRORES
65      valido = 1 #MARCA DE VALIDEZ DE GRUPO DE DATOS RECOLECTADOS
66
67      while len(datos) < self.cantidad_recibida or valido != 1:
68        # VERIFICACION DE TIEMPO SI ES MAYOR QUE LA ESPERA MAXIMA
              REINICIA PROCESO
69        if time.time() - tiempo_inicio > espera_maxima:
70          fprint("[WARN] Tiempo excedido     reenviando posicion...")
71          self.ser.reset_input_buffer()
72          ################################################
73          self.funcion_posicion_actual()# FUNCION QUE ACTUALIZA LA
                POSICION ACTUAL REAL FISICA
74          self.ser.write((self.coefcientes_string + "\n").encode()) #
                ENVIO DE KPP KPV KIV POR SERIAL
75          time.sleep(0.001)
76          self.ser.write((str(self.posicion_actual) + "\n").encode())#
                ENVIO DE NUEVA POSICION
77          print(f"[TX] Reenv o -> Nueva posicion: {self.posicion_actual
                }")
78          datos.clear() #REINICIO DE datos PARA NUEVA MUESTRA
79          tiempo_inicio = time.time()  # REINICIO DEL LIMITE DE TIEMPO
                POR MUESTRA
80          valido = 1 #INDICACION DE QUE LA MUESTRA ES VALIDA
81          continue
82        # LECTURA NORMAL DEL PUERTO SERIAL
83        line = self.ser.readline().decode(errors="ignore").strip()
84        if not line:
85          continue
86        try:
87          value = float(line)  # CONVERSION DE DATO EN VALOR NUMERICO
88        # DETECCION DE ERROR DE  SALTO > 0.02 EN 2 VALORES CONSECUTIVOS
              PARA ELLIMINAR MUESTRA
89        if len(datos) > 1:  # SE NECESITA AL MENOS 2 VALORES PREVIOS
              PARA LA PRUEBA
90          # VERIFICACION DE SI HUBO SALTOS CONSECUTIVOS MAYORES A > 0.02
91          salto_actual = abs(value - datos[-2])
92          salto_anterior = abs(datos[-1] - datos[-2])
93
```

```
94          if salto_actual > 0.02 and salto_anterior > 0.01:
95            print(f"[WARN] Dos saltos consecutivos detectados:")
96            valido =0 #SI HAY ERROR SE DECLARA INVALIDA LA MUESTRA DE
                 DATOS
97
98        datos.append(value)  # GUARDANDO DATOS
99        except:
100         print(f"[WARN] L nea inv lida ignorada: {line}")
101         continue
102         print(f"[RX] {len(datos)} muestras capturadas") #CORROBORANDO
               RECIBO DE MUESTRAS TOTAL
103     return np.array(datos)
104   # =====================
105   # FUNCION QUE ACTUALIZA LA POSICION ACTUAL REAL FISICA Y
106   # GARANTIZA VELOCIDAD INICIAL CERCANA A 0
107   # =====================
108   def funcion_posicion_actual(self):
109     while True:
110       param_str = ";".join(map(str, [1, 28.9, 206]))  # ENVIO DE
             PARAMETROS BASE
111       self.ser.write((param_str + "\n").encode())
112       time.sleep(0.1)
113       self.ser.reset_input_buffer() #RESETEO DEL BUFFER DE DATOS PARA
             NUEVA MUESTRA
114       self.ser.write((str("MARCA\n").encode()))  # MARCA DE
             ACTUALIZACION
115       actual_p = self.ser.readline().decode(errors="ignore").strip()
116       if ';' in actual_p:
117         try: # SEPARACION DE DATOS RECIBIDOS EN (ERROR DE POSICION Y
               VELOCIDAD)
118           dato1_str, dato2_str = actual_p.split(';')
119           dato1 = float(dato1_str)
120           dato2 = float(dato2_str)
121           #COMPROBACION SI EL ERROR DE POSICION Y VELOCIDAD NO PASAN
                 LOS LIMITES
122           if (abs(dato1 - self.posicion_actual) > 0.005 or abs(dato2)
                 >0.1):
123             print(f"RECIBIDO  {dato2}  y  {abs(dato1 - self.
                   posicion_actual)}")
124             time.sleep(0.1)
125           # SI SE CUMPLEN LAS CONDICIONES CONTINUA SINO ESPERA A QUE
                 EL SISTEMA SE ESTABILICE
126           else:
127             break
128         except ValueError:
129           time.sleep(0.01)
130           continue
131     self.valor_referencia = dato1  # ASIGNANDO VALOR DE REFERENCIA
132     self.posicion_actual = dato1 + 3.14  # ASIGNANDO POSICION OBJETIVO
          ACTUAL
133   #FUNCION AUXILIAR PARA EVITAR SOBREIMPULSO EXCESIVO E INTERFERENCIA
        ENTRE MUESTRAS
134   def funcion_auxiliar(self):
135     param_str = ";".join(map(str, [1,28.9,206])) #ENVIO DE PARAMETROS
          BASE
136     self.ser.write((param_str + "\n").encode())
137     time.sleep(0.01)
138     self.ser.write((str(self.posicion_actual) + "\n").encode())
139     print(param_str)
140     time.sleep(2.5) # ESPERA, PARA EVITAR RECEPCION DE DATOS DE
          INESTABILIDAD
141     self.ser.reset_input_buffer()
142   # =====================
143   # FUNCION QUE EVITA CAMBIOS BRUSCOS CAUSADOS POR RUIDOS DE
```

```
    COMUNICACION SERIAL
# ====================
def suavizar_saltos(self, arr, umbral=1):
  if len(arr) <= 1: # SE NECESITA MAS DE UN DATOS PARA COMPARAR
    return arr.copy()
  # CREA UNA COPIA PARA NO MODIFICAR EL ORIGINAL
  suavizado = arr.copy()
  # RECORRE LOS VALORES DE MUESTRA RECOLECTADOS Y EVITA SALTOS MAYORES AL UMBRAL
  for i in range(1, len(suavizado)):
    diferencia = abs(suavizado[i] - suavizado[i - 1])
    if diferencia > umbral:
      suavizado[i] = suavizado[i - 1]
      print(f" Salto corregido en indice {i}: {arr[i - 1]:.3f}     {arr[i]:.3f} (diff: {diferencia:.3f})")
  return suavizado
# ====================
# Funcion objetivo, recibe los datos y usa la funcion ITAE para dar puntajes
# ====================
def objective_function(self, x):
  # Escalar coeficientes para usarlos sin importar sus diferencias de valores absolutos
  coeficientes_escalados = x.copy() # copiando valores para evitar cambios inconvenientes
  if len(coeficientes_escalados) >= 2: #PARA PRUEBAS CON DOS O 3 COEFICIENTES
    coeficientes_escalados[0] = coeficientes_escalados[0] * 5#VALOR P
    coeficientes_escalados[1] = coeficientes_escalados[1] * 34.1# VALOR KPV
  if len(coeficientes_escalados) >= 3:
    coeficientes_escalados[2] = coeficientes_escalados[2] * 828# VALOR KIV

  # ACTUALIZACION DE COEFICIENTES EN EL MCU PARA LA PRUEBA ACTUAL
  param_str = ";".join(map(str, coeficientes_escalados))
  self.coefcientes_string=param_str
  self.ser.write((self.coefcientes_string + "\n").encode())
  print(f"[PSO] Enviando parametros: {param_str}")
  time.sleep(0.01)

  # MANDAR NUEVA POCICION Y RECEPCIONAR DATOS DE RESPUESTA ESCALON
  capturados = self.captura_de_datos(cmd="3")
  # SUAVISANDO DATOS
  capturados = self.suavizar_saltos(capturados, umbral=1)
  marca = 0 #MARCA PARA SABER SI SE SUPERARON LOS UMBRALES
  umbral_alto = 3.768 #UMBRAL DE SOBREIMPULSO  120% DE 3.14
  umbral_bajo = 2.5132 #UMBRA BAJO DE 80% DE 3.14
  supero_umbral_alto = False
  supero_umbral_bajo = False
  #SE DETERMINA SI LOS VALORES RECIBIDOS PASARON EL UMBRAL ALTO, DE SER ASI, SE EJECUTA
  #LA FUNCION AUXILIAR
  for valor in capturados:
    try:
      diferencia = abs(valor - self.valor_referencia)
      if diferencia > umbral_alto: #EXPLORA LOS DATOS PARA VER SI SE CRUZA LOS UMBRALES
        print(f"[INFO] Supero el umbral alto ({diferencia:.4f} > {umbral_alto})")
        supero_umbral_alto = True
        try:
          self.funcion_auxiliar()#USA LA FUNCION AUXILIAR PARA
```

```
                ESTABILIZAR EL SISTEMA
195             marca = 1 #MARCA PARA SABER QUE SE CRUZO EL UMBRAL ALTO
196             print("[INFO] Funcion ejecutada por umbral alto.")
197           except Exception as e:
198             print(f"[ERROR] en funcion_auxiliar (umbral alto): {e}")
199           break
200         elif diferencia > umbral_bajo: #EVALUACION DEL UMBRAL BAJO
201           supero_umbral_bajo = True  # REGISTRO DE QUE SE SUPERO EL
                UMBRAL BAJO
202       except Exception as e:
203         print(f"[WARN] Error procesando valor {valor}: {e}")
204
205     # SI NO SE PASARON LOS UMBRALES SE EJECUTA LA FUNCION AUXILIAR
206     if not supero_umbral_alto and not supero_umbral_bajo:
207       print(
208       f"[INFO] No se supero ni {umbral_bajo} ni {umbral_alto},
            ejecutando funcion por condici n de seguridad.")
209       try:
210         self.funcion_auxiliar()
211         marca = 1
212         print("[INFO] Funcion ejecutada por no alcanzar ningun umbral.
              ")
213       except Exception as e:
214         print(f"[ERROR] en funcion_auxiliar (ningun umbral): {e}")
215
216     # DEFINIR NUEVA REFERERNCIA PARA EL CALCULO ITAE
217     ref = self.posicion_actual
218
219     #HALLANDO PUNTAJE  POR FUNCION ITAE Y REGRESANDO EL VALOR
220     n = len(capturados)
221     t = np.arange(n)  # Tiempos discretos: 0, 1, 2, ..., n-1 para
          funcion ITAE
222     #PUNTAJE CON FUNCION ITAE
223     puntaje = np.sum(t/20 * abs(capturados - ref)) #20 ES UN VALOR
          ARBITRARIO PARA EVITAR NROS EXTENSOS
224
225     print(f"[PSO] puntaje={puntaje}")
226     return puntaje, abs(capturados - ref) # ENVIDO DE PUNTAJE Y DATOS
227
228   # =====================
229   # PSO FUNCION PRINCIPAL
230   # =====================
231   def run(self):
232     w, c1, c2 = self.W, self.C1, self.C2 #0.5, 2, 0.5  # PARAMETROS
          USADOS
233     history = []  # PARA GUARDAR DATOS DE LAS PARTICULAS
234
235     self.mejor_curve_por_iter = []# MEJOR CURVA DE CADA ITERACION
236
237     for it in range(self.max_iters): #BUCLE PRINICIPAL DE ITERACION O
          GENERACIONES EN TOTAL
238       print(f"\n=== Iteracion {it+1}/{self.max_iters} ===")
239       datos_de_iteracion = []  # CAPTURAS DE ITERACION ACTUAL
240       mejor_puntaje_iter = float("inf")#COMPARADORES PARA ALMACENAR
            MEJOR ITERACION
241       mejor_captura_iter = None
242
243       for i in range(self.num_particulas): #BUCLE PARA CADA PARTICULA
            POR GENERACION
244         # CALCULANDO PUNTAJES INICIALES DE PARTICULAS ALEATORIAS
245         puntaje, capturados = self.objective_function(self.posiciones[
              i])
246         # GUARDADOS DE DATOS PARA GRAFICAS POSTERIORES
247         datos_de_iteracion.append((i, capturados))
```

```
248         # Registrar mejor curva de ESTA iteracion
249         if puntaje < mejor_puntaje_iter:
250           mejor_puntaje_iter = puntaje
251           mejor_captura_iter = capturados
252         # ACTUALIZANDO MEJOR PUNTAJE DE SER NECESARIO
253         if puntaje < self.personal_best_puntajes[i]:
254           self.personal_best_puntajes[i] = puntaje
255           self.personal_best[i] = self.posiciones[i].copy()
256         # ACTUALIZANDO MEJOR PUNTAJE GENERAL O GLOBAL DE SER NECESARIO
257         if puntaje < self.global_mejor_puntaje:
258           self.global_mejor_puntaje = puntaje
259           self.global_best = self.posiciones[i].copy()
260       # GUARDANDO PARA FUTURAS GRAFICAS
261       history.append(datos_de_iteracion)
262       self.mejor_puntaje_historico.append(self.global_mejor_puntaje)
263       self.mejor_curve_por_iter.append(mejor_captura_iter)  # GUARDADO
              DE MEJORES PUNTAJES Y MEJOR CURVA
264       self.posiciones_historico.append(self.posiciones.copy())
265
266       # ACTUALIZACION DE POSICIONES Y VELOCIDAD SEGUN ALGORITMO DE PSO
267       # ========================================================
268       # SELECCION DE COEFICIENTES ALEATORIOS PARA EL ENJAMBRE SEGUN
              ALGORITMO PSO
269       # ========================================================
270       r1 = np.random.rand(self.num_particulas, self.dimensions)
271       r2 = np.random.rand(self.num_particulas, self.dimensions)
272       # ========================================================
273       # GANANCIA COGINITIVA DEL MEJOR PUNTAJE PERSONAL Y MEJOR GLOBAL
              SEGUN ALGORITMO PSO
274       # ========================================================
275       cognitive = c1 * r1 * (self.personal_best - self.posiciones)
276       social = c2 * r2 * (self.global_best - self.posiciones)
277       # ========================================================
278       # ACTUALIZACION DE VELOCIDAD Y POSICION SEGUN ALGORITMO PSO
279       # ========================================================
280       self.velocities = w * self.velocities + cognitive + social
281       self.posiciones += self.velocities
282
283       # ACOTANDO LAS POSICIONES AL AREA DE BUSQUEDA
284       self.posiciones = np.clip(self.posiciones, self.bounds[0], self.
              bounds[1])
285     # ================================
286     # FINAL
287     # ================================
288     #INDICADORES DE FINALIZACION Y MEJORES PARAMETROS OBTENIDOS
289     with open(f"{CARPETA_RESULTADOS}/COEFICIENTES.txt", "w") as f:
290       f.write(f"{self.global_best}\n")
291       f.write(f"{self.global_mejor_puntaje}\n")
292     print("\n===PSO TERMINADO ===")
293     print("Mejor posicion encontrada:", self.global_best)
294     print("Mejor puntaje:", self.global_mejor_puntaje)
295
296
297
298     # ====================
299     # Graficar evolucion del puntaje durante las generaciones
300     # ====================
301     with open(f"{CARPETA_RESULTADOS}/MEJOR HISTORICO.txt", "w") as f:
302       for valor in self.mejor_puntaje_historico:
303         f.write(f"{valor}\n")
304     plt.figure(figsize=(8, 5))
305     plt.plot(self.mejor_puntaje_historico, marker="o")
306     plt.title("Evolucion del mejor puntaje")
307     plt.xlabel("Iteracion")
```

```
308     plt.ylabel("puntaje (error penalizado)")
309     plt.grid(True)
310     # GUARDANDO EN DOS FORMATOS
311     plt.savefig(f"{CARPETA_RESULTADOS}/MEJORES_PUNTAJES.eps", format='
            eps', dpi=300, bbox_inches='tight')
312     plt.savefig(f"{CARPETA_RESULTADOS}/MEJORES_PUNTAJES.png", format='
            png', dpi=300, bbox_inches='tight')
313     plt.show()
314
315     # ====================
316     # Graficar trayectorias en 2D (3 graficas para 3 dimensiones
            convinadas)
317     # La grafica se realiza de este modo para facilitar la
            visualizacion
318     # ====================
319     if self.dimensions == 3:
320       positions_hist = np.array(self.posiciones_historico)
321       step_marker = 1  # puedes poner 1 si quieres todos los puntos
322
323       # CREAR FIGURA CON 3 SUBPLOTS
324       fig, axes = plt.subplots(1, 3, figsize=(15, 4))
325
326       # Grafico 1: Dimensi n 1 vs Dimensi n 2
327       ax1 = axes[0]
328       for p in range(self.num_particulas):
329         xs = positions_hist[:, p, 0]
330         ys = positions_hist[:, p, 1]
331         ax1.plot(xs, ys, alpha=0.6, linewidth=1, label=f"Particula {p
              + 1}")
332         # Agregar puntos
333         ax1.scatter(xs[::step_marker], ys[::step_marker], s=3)
334       ax1.scatter(self.global_best[0], self.global_best[1],
335       c="red", marker="*", s=200, label="Global Best")
336       ax1.set_title("Kpp_normalizado vs Kpi_normalizado")
337       ax1.set_xlabel("Kpp_normalizado")
338       ax1.set_ylabel("Kpi_normalizado")
339       ax1.grid(True, alpha=0.6)
340
341       # Grafico 2: Dimensi n 2 vs Dimensi n 3
342       ax2 = axes[1]
343       for p in range(self.num_particulas):
344         ys = positions_hist[:, p, 1]
345         zs = positions_hist[:, p, 2]
346         ax2.plot(ys, zs, alpha=0.6, linewidth=1)
347         # Agregar puntos
348         ax2.scatter(ys[::step_marker], zs[::step_marker], s=3)
349       ax2.scatter(self.global_best[1], self.global_best[2],
350       c="red", marker="*", s=200)
351       ax2.set_title("Kpi_normalizado vs Kvi_normalizado")
352       ax2.set_xlabel("Kpi_normalizado")
353       ax2.set_ylabel("Kvi_normalizado")
354       ax2.grid(True, alpha=0.6)
355
356       # Grafico 3: Dimensi n 1 vs Dimensi n 3
357       ax3 = axes[2]
358       for p in range(self.num_particulas):
359         xs = positions_hist[:, p, 0]
360         zs = positions_hist[:, p, 2]
361         ax3.plot(xs, zs, alpha=0.6, linewidth=1)
362         # Agregar puntos
363         ax3.scatter(xs[::step_marker], zs[::step_marker], s=3)
364       ax3.scatter(self.global_best[0], self.global_best[2],
365       c="red", marker="*", s=200)
366       ax3.set_title("Kpp_normalizado vs Kvi_normalizado")
```

```
367         ax3.set_xlabel("Kpp_normalizado")
368         ax3.set_ylabel("Kvi_normalizado")
369         ax3.grid(True, alpha=0.6)
370 
371 
372         plt.tight_layout()
373         plt.savefig(f"{CARPETA_RESULTADOS}/TRAYECTORIAS_PARTICULAS.eps",
                format='eps', dpi=300, bbox_inches='tight')
374         plt.savefig(f"{CARPETA_RESULTADOS}/TRAYECTORIAS_PARTICULAS.png",
                format='png', dpi=300, bbox_inches='tight')
375         plt.show()
376         plt.close('all')
377 
378         plt.figure(figsize=(12, 8))
379         # # ======================
380         # # GRAFICANDO CURVAS CON DIFERENTES COLORES PARA DISTINGUILAS
381         # # ======================
382         colors = plt.cm.jet(np.linspace(0, 1, len(history)))
383 
384         idx_best_gen = np.argmin(self.mejor_puntaje_historico)
385         best_curve = self.mejor_curve_por_iter[idx_best_gen]
386 
387         # GUARDAR LA MEJOR CURVA EN UN ARCHIVO EXTERNO
388         with open(f"{CARPETA_RESULTADOS}/MEJOR_CURVA.txt", "w") as f:
389           for valor in best_curve:
390             f.write(f"{valor}\n")
391 
392         for idx, iteration in enumerate(history):
393           color = colors[idx]
394           for i, capturados in iteration:
395             plt.plot(capturados, color=color, alpha=0.6, linewidth=0.6)
396 
397         plt.plot(best_curve, color="black", linewidth=3, label="Mejor
              curva global")
398 
399         plt.xlabel("Indice de muestra")
400         plt.ylabel("Valor")
401         plt.title("Evolucion de todas las particulas (Mejor curva GLOBAL
              resaltada)")
402         plt.legend()
403         plt.grid(True)
404         plt.savefig(f"{CARPETA_RESULTADOS}/CURVAS_COMPORTAMIENTO.eps",
              format='eps', dpi=300, bbox_inches='tight')
405         plt.savefig(f"{CARPETA_RESULTADOS}/CURVAS_COMPORTAMIENTO.png",
              format='png', dpi=300, bbox_inches='tight')
406         plt.show()
407 
408     return self.global_best, self.global_mejor_puntaje
409 
410 
411 # LLAMADO A FUNCION PRINCIPAL Y DEFINICION DE CARACTERISTICAS
      PRINCIPALES
412 if __name__ == "__main__":
413   system = SerialDE(port="COM10", baudrate=1000000, cantidad_recibida
        =2000, num_particulas=30,
414   dimensions=3, bounds=(0, 1.5), max_iters=50, W=0.5,C1=2,C2=0.5)#20
        10    30   20 #C1 PERSONAL
415   CARPETA_RESULTADOS = "PSO_30_50_0.5_2_0.5_test11"#"PSO_50_50_0.5_2_0
        .5_test6"  # Cambia esto
416   os.makedirs(f"{CARPETA_RESULTADOS}", exist_ok=True)
417   best_pos, best_puntaje = system.run()
```

Listing 4.2 *Funcion base de DE*

```
## Optimizacion de control de posicion con algoritmo de DE
import serial
import time
import numpy as np
import matplotlib.pyplot as plt
import os

class SerialDE: #INICIO DE LA CLASE CON PARAMETROS TENPORALES
  def __init__(self, port="COM10", baudrate=1000000, cantidad_recibida=100,
    num_particulas=5, dimensions=2, bounds=(0, 1), max_iters=3, F=0.0, CR=0.0):
    # CONFIGURACION DE ALGORITMO DE
    self.F = F
    self.CR = CR
    #  Configuracion de Serial, BAUDIOS , PORT, ETC.
    self.port = port
    self.baudrate = baudrate
    self.cantidad_recibida = cantidad_recibida #INDICADOR DE CANTIDAD DE MUESTRAS
    self.valor_referencia = 3.14  #VALOR DE REFERENCIA USADO PARA HALLAR EL ERROR DE POSICION
    self.posicion_actual = 0.0  #VALOR DE ANGULO DE POSICION INICIAL EN RADIANES

    #LISTAS PARA GRUARDADO DE DATOS Y GRAFICOS FINALES
    self.mejor_curve_por_iter = []
    self.mejor_index_por_iter = []

    try:
      # CONEXION INICIAL DE SERIAL Y ERROR DE CONEXION SI NO DISPONIBLE
      self.ser = serial.Serial(self.port, self.baudrate, timeout=1)
      print(f"[INFO] Serial abierto en {self.port} @ {self.baudrate}")
    except Exception as e:
      raise RuntimeError(f"No se pudo abrir puerto serial: {e}")

    # CONFIGURACION DE PARAMETROS DE
    self.num_particulas = num_particulas  # TOTAL DE PARTICULAS PARA ITERACIONES
    self.dimensions = dimensions  # DIMENSIONES TOTALES O NUMERO DE PARAMETROS (3)
    self.bounds = bounds  # (min, max) LIMITES DE VALORES PARA LOS PARAMETROS
    self.max_iters = max_iters

    # INICIALIZACION DE POBLACION
    # VALORES ALEATORIOS DE PARTICULAS
    self.posiciones = np.random.uniform(bounds[0], bounds[1],(num_particulas, dimensions))
    # AJUSTANDO LOS VALORES DE LAS PARTICULAS A LOS LIMITES
    self.posiciones = np.clip(self.posiciones, bounds[0],bounds[1])
    # INICIANDO VALORES DE VELOCIDADES DE PARTICULAS
    self.velocities = np.zeros((num_particulas, dimensions))
    # ASIGNACION DE MEJORES VALORES INDIVIDUALES
    self.personal_best = self.posiciones.copy()
    # PUNTAJE MUY ALTO INICIAL PARA QUE PUEDA SER FACILMENTE REEMPLAZADO
    self.personal_best_puntajes = np.full(num_particulas,np.inf)
    # DECLARACION PERO NO DEFINICION DE LA MEJOR PARTICULA EN GENERAL
    self.global_best = None
    # PUNTAJE MUY ALTO DE MEJOR PARTICULA INICIAL
```

```
52      self.global_mejor_puntaje = np.inf
53
54      # HISTORIAL DE DATOS PARA GRAFICAR LA EVOLUCION DE LA OPTIMIZACION
55      self.mejor_puntaje_historico = []  # GUARDANDO PUNTAJES Y
            POSICIONES PARA GRAFICAS FINALES
56      self.posiciones_historico = []
57      self.coefcientes_string = ""
58
59    #FUNCION DE CAPTURA DE DATOS PARA CADA ELEMENTO DE POBLACION PROBADA
60    def captura_de_datos(self, cmd="2"):
61      self.ser.reset_input_buffer()  # LIMPIANDO ENTRADA DE SERIAL DE
            DATOS EXTRAS NO DESEADOS
62      ########################################################
63      self.funcion_posicion_actual() # FUNCION QUE ACTUALIZA LA POSICION
            ACTUAL REAL FISICA
64      self.ser.write((self.coefcientes_string + "\n").encode())#
            ACTUALIZANDO PARAMETROS KPP,KIV,PKV
65      time.sleep(0.001)
66      self.ser.write((str(self.posicion_actual) + "\n").encode())  #
            ENVIANDO NUEVA POCICION A MCU
67      print(self.posicion_actual)
68      print(f"[TX] {cmd} -> esperando {self.cantidad_recibida} muestras"
            )  # ESPERA DE MUESTRAS
69      datos = []  # DATOS ALMACENADOS PARA UNA SOLA PARTICULA
70      tiempo_inicio = time.time()  # TIEMPO PARA EVITAR ERRORES DE
            EXCESIVO TIEMPO DE RECEPCION
71      espera_maxima = 2.5  # ESPERA EN SEGUNDOS PARA EVITAR ERRRORES
72      valido = 1 #MARCA DE VALIDEZ DE GRUPO DE DATOS RECOLECTADOS
73
74      while len(datos) < self.cantidad_recibida or valido != 1:
75        # VERIFICACION DE TIEMPO SI ES MAYOR QUE LA ESPERA MAXIMA
              REINICIA PROCESO
76        if time.time() - tiempo_inicio > espera_maxima:
77          print("[WARN] Tiempo excedido     reenviando posicion...")
78          self.ser.reset_input_buffer()
79          ########################################################
80          self.funcion_posicion_actual()# FUNCION QUE ACTUALIZA LA
                POSICION ACTUAL REAL FISICA
81          self.ser.write((self.coefcientes_string + "\n").encode())#
                ENVIO DE KPP KPV KIV POR SERIAL
82          time.sleep(0.001)
83          self.ser.write((str(self.posicion_actual) + "\n").encode())#
                ENVIO DE NUEVA POSICION
84          print(f"[TX] Reenv o -> Nueva posicion: {self.posicion_actual
                }")
85          datos.clear() #REINICIO DE datos PARA NUEVA MUESTRA
86          tiempo_inicio = time.time()  # REINICIO DEL LIMITE DE TIEMPO
                POR MUESTRA
87          valido = 1 #INDICACION DE QUE LA MUESTRA ES VALIDA
88          continue
89
90        # LECTURA NORMAL DEL PUERTO SERIAL
91        line = self.ser.readline().decode(errors="ignore").strip()
92        if not line:
93          continue
94        try:
95          value = float(line)  # CONVERSION DE DATO EN VALOR NUMERICO
96
97          # DETECCION DE ERROR DE  SALTO > 0.02 EN 2 VALORES
                CONSECUTIVOS PARA ELLIMINAR MUESTRA
98          if len(datos) > 1:  # SE NECESITA AL MENOS 2 VALORES PREVIOS
                PARA LA PRUEBA
99            # VERIFICACION DE SI HUBO SALTOS CONSECUTIVOS MAYORES A >
                  0.02
```

```
100             salto_actual = abs(value - datos[-2])
101             salto_anterior = abs(datos[-1] - datos[-2])
102 
103             if salto_actual > 0.02 and salto_anterior > 0.01:
104               print(f"[WARN] Dos saltos consecutivos detectados:")
105               valido = 0 #SI HAY ERROR SE DECLARA INVALIDA LA MUESTRA DE
                      DATOS
106 
107           datos.append(value)  # GUARDANDO DATOS
108         except:
109           print(f"[WARN] L nea inv lida ignorada: {line}")
110           continue
111       print(f"[RX] {len(datos)} muestras capturadas")  # CORROBORANDO
          RECIBO DE MUESTRAS TOTAL
112 
113       return np.array(datos)
114 
115     # =====================
116     # FUNCION QUE ACTUALIZA LA POSICION ACTUAL REAL FISICA Y
117     # GARANTIZA VELOCIDAD INICIAL CERCANA A 0
118     # =====================
119     def funcion_posicion_actual(self):
120       while True:
121         param_str = ";".join(map(str, [1, 28.9, 206]))  # ENVIO DE
              PARAMETROS BASE
122         self.ser.write((param_str + "\n").encode())
123         time.sleep(0.1)
124         self.ser.reset_input_buffer() #RESETEO DEL BUFFER DE DATOS PARA
              NUEVA MUESTRA
125         self.ser.write((str("MARCA\n").encode()))  # MARCA DE
              ACTUALIZACION
126         actual_p = self.ser.readline().decode(errors="ignore").strip()
127         if ';' in actual_p:
128           try:# SEPARACION DE DATOS RECIBIDOS EN (ERROR DE POSICION Y
                 VELOCIDAD)
129             dato1_str, dato2_str = actual_p.split(';')
130             dato1 = float(dato1_str)
131             dato2 = float(dato2_str)
132             # COMPROBACION SI EL ERROR DE POSICION Y VELOCIDAD NO PASAN
                 LOS LIMITES
133             if (abs(dato1 - self.posicion_actual) > 0.005 or abs(dato2)
                 > 0.1):
134               print(f"RECIBIDO  {dato2}  y  {abs(dato1 - self.
                    posicion_actual)}")
135               time.sleep(0.1)
136             # SI SE CUMPLEN LAS CONDICIONES CONTINUA SINO ESPERA A QUE
                 EL SISTEMA SE ESTABILICE
137             else:
138               break
139           except ValueError:
140             time.sleep(0.01)
141             continue
142       self.valor_referencia = dato1  # ASIGNANDO VALOR DE REFERENCIA
143       self.posicion_actual = dato1 + 3.14  # ASIGNANDO POSICION OBJETIVO
              ACTUAL
144 
145     # FUNCION AUXILIAR PARA EVITAR SOBREIMPULSO EXCESIVO E INTERFERENCIA
          ENTRE MUESTRAS
146     def funcion_auxiliar(self):
147       param_str = ";".join(map(str, [1, 28.9, 206]))  #ENVIO DE
            PARAMETROS BASE
148       self.ser.write((param_str + "\n").encode())
149       time.sleep(0.01)
150       self.ser.write((str(self.posicion_actual) + "\n").encode())
```

```
151     print(param_str)
152     time.sleep(2.5)  # ESPERA, PARA EVITAR RECEPCION DE DATOS DE
            INESTABILIDAD
153     self.ser.reset_input_buffer()
154
155   # =====================
156   # FUNCION QUE EVITA CAMBIOS BRUSCOS CAUSADOS POR RUIDOS DE
          COMUNICACION SERIAL
157   # =====================
158   def suavizar_saltos(self, arr, umbral=1):
159     if len(arr) <= 1: # SE NECESITA MAS DE UN DATOS PARA COMPARAR
160       return arr.copy()
161     # CREA UNA COPIA PARA NO MODIFICAR EL ORIGINAL
162     suavizado = arr.copy()
163     # RECORRE LOS VALORES DE MUESTRA RECOLECTADOS Y EVITA SALTOS
            MAYORES AL UMBRAL
164     for i in range(1, len(suavizado)):
165       diferencia = abs(suavizado[i] - suavizado[i - 1])
166       if diferencia > umbral:
167         suavizado[i] = suavizado[i - 1]
168         print(f" Salto corregido en indice {i}: {arr[i - 1]:.3f}     {
              arr[i]:.3f} (diff: {diferencia:.3f})")
169     return suavizado
170
171   # =====================
172   # Funcion objetivo, recibe los datos y usa la funcion ITAE para dar
          puntajes
173   # =====================
174   def objective_function(self, x):
175     # Escalar coeficientes para usarlos sin importar sus diferencias
            de valores absolutos
176     coeficientes_escalados = x.copy()  # copiando valores para evitar
            cambios inconvenientes
177     if len(coeficientes_escalados) >= 2:  # PARA PRUEBAS CON DOS O 3
            COEFICIENTES
178       coeficientes_escalados[0] = coeficientes_escalados[0] * 5  #
              VALOR P
179       coeficientes_escalados[1] = coeficientes_escalados[1] * 34.1  #
              VALOR KPV
180     if len(coeficientes_escalados) >= 3:
181       coeficientes_escalados[2] = coeficientes_escalados[2] * 828  #
              VALOR KIV
182
183     # ACTUALIZACION DE COEFICIENTES EN EL MCU PARA LA PRUEBA ACTUAL
184     param_str = ";".join(map(str, coeficientes_escalados))
185     self.coefcientes_string = param_str
186     self.ser.write((self.coefcientes_string + "\n").encode())
187     print(f"[PSO] Enviando parametros: {param_str}")
188     time.sleep(0.01)
189
190     # MANDAR NUEVA POCICION Y RECEPCIONAR DATOS DE RESPUESTA ESCALON
191     capturados = self.captura_de_datos(cmd="3")
192     # SUAVISANDO DATOS
193     capturados = self.suavizar_saltos(capturados, umbral=1)
194     marca = 0  # MARCA PARA SABER SI SE SUPERARON LOS UMBRALES
195     umbral_alto = 3.768  # UMBRAL DE SOBREIMPULSO  120% DE 3.14
196     umbral_bajo = 2.5132  # UMBRA BAJO DE 80% DE 3.14
197     supero_umbral_alto = False
198     supero_umbral_bajo = False
199     # SE DETERMINA SI LOS VALORES RECIBIDOS PASARON EL UMBRAL ALTO, DE
            SER ASI, SE EJECUTA
200     # LA FUNCION AUXILIAR
201     for valor in capturados:
202       try:
```

```
203         diferencia = abs(valor - self.valor_referencia)
204           if diferencia > umbral_alto:#EXPLORA LOS DATOS PARA VER SI
                SE CRUZA LOS UMBRALES
205           print(f"[INFO] Supero el umbral alto ({diferencia:.4f} > {
                umbral_alto})")
206           supero_umbral_alto = True
207           try:
208             self.funcion_auxiliar()#USA LA FUNCION AUXILIAR PARA
                  ESTABILIZAR EL SISTEMA
209             marca = 1 #MARCA PARA SABER QUE SE CRUZO EL UMBRAL ALTO
210             print("[INFO] Funcion ejecutada por umbral alto.")
211           except Exception as e:
212             print(f"[ERROR] en funcion_auxiliar (umbral alto): {e}")
213           break
214         elif diferencia > umbral_bajo: #EVALUACION DEL UMBRAL BAJO
215           supero_umbral_bajo = True  # REGISTRO DE QUE SE SUPERO EL
                UMBRAL BAJO
216       except Exception as e:
217         print(f"[WARN] Error procesando valor {valor}: {e}")
218
219     # SI NO SE PASARON LOS UMBRALES SE EJECUTA LA FUNCION AUXILIAR
220     if not supero_umbral_alto and not supero_umbral_bajo:
221       print(
222       f"[INFO] No se supero ni {umbral_bajo} ni {umbral_alto},
            ejecutando funcion por condici n de seguridad.")
223       try:
224         self.funcion_auxiliar()
225         marca = 1
226         print("[INFO] Funcion ejecutada por no alcanzar ningun umbral.
              ")
227       except Exception as e:
228         print(f"[ERROR] en funcion_auxiliar (ningun umbral): {e}")
229
230     # DEFINIR NUEVA REFERERNCIA PARA EL CALCULO ITAE
231     ref = self.posicion_actual
232
233     # HALLANDO PUNTAJE  POR FUNCION ITAE Y REGRESANDO EL VALOR
234     n = len(capturados)
235     t = np.arange(n)  # Tiempos discretos: 0, 1, 2, ..., n-1 para
          funcion ITAE
236     # PUNTAJE CON FUNCION ITAE
237     puntaje = np.sum(t / 20 * abs(capturados - ref)) #20 ES UN VALOR
          ARBITRARIO PARA EVITAR NROS EXTENSOS
238
239     print(f"[PSO] puntaje={puntaje}")
240     return puntaje, abs(capturados - ref)  # ENVIDO DE PUNTAJE Y DATOS
241
242   # ======================
243   # DE FUNCION PRINCIPAL
244   # ======================
245   def run_DE(self):
246     F = self.F  # Factor de mutacion 0.2
247     CR = self.CR  # Probabilidad de cruzamiento 0.5
248
249     history = [] # PARA GUARDAR DATOS DE LAS PARTICULAS
250     print("\n=== Iniciando Evolucion Diferencial (DE) ===\n")
251
252     # EVALUACION INICIAL DE PARAMETROS ALEATORIOS
253     puntajes = np.zeros(self.num_particulas)
254     for i in range(self.num_particulas):
255       puntaje, capt = self.objective_function(self.posiciones[i])
256       puntajes[i] = puntaje
257       self.personal_best[i] = self.posiciones[i].copy() #ACTUALIZACION
            DE MEJORES PUNTAJES
```

```
258         self.personal_best_puntajes[i] = puntaje
259
260         if puntaje < self.global_mejor_puntaje:#ACTUALIZACION DE MEJOR
                GLOBAL
261           self.global_mejor_puntaje = puntaje
262           self.global_best = self.posiciones[i].copy()
263
264       # ALMACENAMIENTO PARA GRAFICOS POSTERIORES
265       self.mejor_puntaje_historico.append(self.global_mejor_puntaje)
266       self.posiciones_historico.append(self.posiciones.copy())
267
268       # ================================
269       # BUCLE PRINCIPAL DE DE
270       # ================================
271       for it in range(self.max_iters): #BUCLE DE GENERACIONES TOTALES
272         print(f"\n=== Iteracion {it + 1}/{self.max_iters} ===")
273
274         datos_de_iteracion = []
275
276         # NUEVO ALMACENANDO MEJOR CURVA DE LA ITERACION
277         mejor_puntaje_iter = float("inf")
278         mejor_curve_iter = None
279         mejor_idx_iter = None
280
281         for i in range(self.num_particulas):
282
283           # ================================================
284           # MUTACION EVITANDO USAR LA PARTICULA DE ITERACION ACTUAL
                SEGUN ALGORITMO DE
285           # ================================================
286           indices = list(range(self.num_particulas))
287           indices.remove(i)
288           a, b, c = np.random.choice(indices, 3, replace=False) #
                ESCOGIENDO ALEATORIAMENTE
289
290           Xa = self.posiciones[a]
291           Xb = self.posiciones[b]
292           Xc = self.posiciones[c]
293
294           V = Xa + F * (Xb - Xc) #MUTACION
295
296           # ================================================
297           # CRUZAMIENTO SEGUN ALGORITMO DE
298           # ================================================
299           U = self.posiciones[i].copy()
300           j_rand = np.random.randint(self.dimensions)
301
302           for j in range(self.dimensions):
303             if np.random.rand() < CR or j == j_rand:
304               U[j] = V[j]
305
306           U = np.clip(U, self.bounds[0], self.bounds[1])
307
308           # ================================================
309           # EVALUACI N DEL VECTOR DE PRUEBA
310           # ================================================
311           puntaje_U, capt_U = self.objective_function(U)
312           puntaje_Xi = puntajes[i]
313
314           # Guardar curva capturada de esta particula
315           datos_de_iteracion.append((i, capt_U))
316           # ================================================
317           # NUEVO, GUARDANDO MEJOR CURVA DE ITERACION
318           # ================================================
```

```
319           if puntaje_U < mejor_puntaje_iter:
320             mejor_puntaje_iter = puntaje_U
321             mejor_curve_iter = capt_U.copy()
322             mejor_idx_iter = i
323           # ================================================
324           # SELECCI N SEGUN ALGORITMO DE
325           # ================================================
326           if puntaje_U < puntaje_Xi:
327             self.posiciones[i] = U
328             puntajes[i] = puntaje_U
329
330             # actualizar personal best
331             self.personal_best[i] = U.copy()
332             self.personal_best_puntajes[i] = puntaje_U
333
334             # actualizar global best
335             if puntaje_U < self.global_mejor_puntaje:
336               self.global_mejor_puntaje = puntaje_U
337               self.global_best = U.copy()
338
339       # ========================================================
340       # GUARDANDO DATOS DE LA ITERACION PARA FUTURAS GRAFICAS
341       # ========================================================
342       history.append(datos_de_iteracion)
343       self.mejor_puntaje_historico.append(self.global_mejor_puntaje)
344       self.posiciones_historico.append(self.posiciones.copy())
345
346       # GUARDAR MEJOR CURVA DE ITERACION ACTUAL
347       self.mejor_curve_por_iter.append(mejor_curve_iter)
348       self.mejor_index_por_iter.append(mejor_idx_iter)
349
350     # ================================
351     # FINAL
352     # ================================
353     # INDICADORES DE FINALIZACION Y MEJORES PARAMETROS OBTENIDOS
354     with open(f"{carpeta_resultados}/COEFICIENTES.txt", "w") as f:
355       f.write(f"{self.global_best}\n")
356       f.write(f"{self.global_mejor_puntaje}\n")
357     print("\n=== DE Terminado ===")
358     print("Mejor posicion encontrada:", self.global_best)
359     print("Mejor puntaje:", self.global_mejor_puntaje)
360
361     # =====================
362     # Graficar evolucion del puntaje durante las generaciones
363     # =====================
364     with open(f"{carpeta_resultados}/MEJOR HISTORICO.txt", "w") as f:
365       for valor in self.mejor_puntaje_historico:
366         f.write(f"{valor}\n")
367
368     plt.figure(figsize=(8, 5))
369     plt.plot(self.mejor_puntaje_historico, marker="o")
370     plt.title("Evolucion del mejor puntaje")
371     plt.xlabel("Iteracion")
372     plt.ylabel("puntaje (error penalizado)")
373     plt.grid(True)
374     plt.savefig(f"{carpeta_resultados}/MEJORES_PUNTAJES.eps", format='
        eps', dpi=300, bbox_inches='tight')
375     plt.savefig(f"{carpeta_resultados}/MEJORES_PUNTAJES.png", format='
        png', dpi=300, bbox_inches='tight')
376     plt.show()
377
378     # =====================
379     # Graficar trayectorias en 2D (3 graficas para 3 dimensiones
        convinadas)
```

```
380     # La grafica se realiza de este modo para facilitar la
            visualizacion
381     # ====================
382     if self.dimensions == 3:
383       positions_hist = np.array(self.posiciones_historico)
384
385       step_marker = 1  # puedes poner 1 si quieres todos los puntos
386       # Crear figura con 3 subplots
387       fig, axes = plt.subplots(1, 3, figsize=(15, 4))
388
389       # Grafico 1: Dimensi n 1 vs Dimensi n 2
390       ax1 = axes[0]
391       for p in range(self.num_particulas):
392         xs = positions_hist[:, p, 0]
393         ys = positions_hist[:, p, 1]
394         ax1.plot(xs, ys, alpha=0.6, linewidth=1, label=f"Particula {p
                + 1}")
395         # Agregar puntos
396         ax1.scatter(xs[::step_marker], ys[::step_marker], s=3)
397       ax1.scatter(self.global_best[0], self.global_best[1],
398       c="red", marker="*", s=200, label="Global Best")
399       ax1.set_title("Kpp_normalizado vs Kpi_normalizado")
400       ax1.set_xlabel("Kpp_normalizado")
401       ax1.set_ylabel("Kpi_normalizado")
402       ax1.grid(True, alpha=0.6)
403
404       # Grafico 2: Dimensi n 2 vs Dimensi n 3
405       ax2 = axes[1]
406       for p in range(self.num_particulas):
407         ys = positions_hist[:, p, 1]
408         zs = positions_hist[:, p, 2]
409         ax2.plot(ys, zs, alpha=0.6, linewidth=1)
410         # Agregar puntos
411         ax2.scatter(ys[::step_marker], zs[::step_marker], s=3)
412         ax2.scatter(self.global_best[1], self.global_best[2],
413       c="red", marker="*", s=200)
414       ax2.set_title("Kpi_normalizado vs Kvi_normalizado")
415       ax2.set_xlabel("Kpi_normalizado")
416       ax2.set_ylabel("Kvi_normalizado")
417       ax2.grid(True, alpha=0.6)
418
419       # Grafico 3: Dimensi n 1 vs Dimensi n 3
420       ax3 = axes[2]
421       for p in range(self.num_particulas):
422         xs = positions_hist[:, p, 0]
423         zs = positions_hist[:, p, 2]
424         ax3.plot(xs, zs, alpha=0.6, linewidth=1)
425         # Agregar puntos
426         ax3.scatter(xs[::step_marker], zs[::step_marker], s=3)
427       ax3.scatter(self.global_best[0], self.global_best[2],
428       c="red", marker="*", s=200)
429       ax3.set_title("Kpp_normalizado vs Kvi_normalizado")
430       ax3.set_xlabel("Kpp_normalizado")
431       ax3.set_ylabel("Kvi_normalizado")
432       ax3.grid(True, alpha=0.6)
433
434       plt.tight_layout()
435       plt.savefig(f"{carpeta_resultados}/TRAYECTORIAS_PARTICULAS.eps",
              format='eps', dpi=300, bbox_inches='tight')
436       plt.savefig(f"{carpeta_resultados}/TRAYECTORIAS_PARTICULAS.png",
              format='png', dpi=300, bbox_inches='tight')
437       plt.show()
438       plt.close('all')
439
```

```
        # ======================
        # GRAFICAR CAPTURAS RESALTANDO LA MEJOR CURVA GLOBAL
        # ======================

        plt.figure(figsize=(12, 8))

        # PALETA DE COLORES
        colors = plt.cm.jet(np.linspace(0, 1, len(history)))

        # 1. ENCONTRAR MEJOR PUNTAJE GLOBAL
        idx_best_gen = np.argmin(self.mejor_puntaje_historico)

        #   TOMANDO LA MEJOR CURVA DE LAS ITERACIONES
        best_curve = self.mejor_curve_por_iter[idx_best_gen]

        # GUARDANDO MEJOR CURVA EN ARCHIVO EXTERNO
        with open(f"{carpeta_resultados}/MEJOR_CURVA.txt", "w") as f:
          for valor in best_curve:
            f.write(f"{valor}\n")

        # GRAFICANDO TODAS LAS CURVAS
        for idx, iteration in enumerate(history):
          color = colors[idx]
          for i, capturados in iteration:
            plt.plot(capturados, color=color, alpha=0.6, linewidth=0.6)

        # RESALTANDO MEJOR CURVA GLOBAL
        plt.plot(best_curve, color="black", linewidth=3, label="Mejor
            curva global")

        # CONFIGURANDO GRAFICO
        plt.xlabel("Indice de muestra")
        plt.ylabel("Valor")
        plt.title("Evolucion de todas las particulas (Mejor curva GLOBAL
            resaltada)")
        plt.legend()
        plt.grid(True)
        # GUARDANDO EN DOS FORMATOS
        plt.savefig(f"{carpeta_resultados}/CURVAS_COMPORTAMIENTO.eps",
            format='eps', dpi=300, bbox_inches='tight')
        plt.savefig(f"{carpeta_resultados}/CURVAS_COMPORTAMIENTO.png",
            format='png', dpi=300, bbox_inches='tight')
        plt.show()
      return self.global_best, self.global_mejor_puntaje

# LLAMADO A FUNCION PRINCIPAL Y DEFINICION DE CARACTERISTICAS
    PRINCIPALES
if __name__ == "__main__":
  system = SerialDE(port="COM10", baudrate=1000000, cantidad_recibida
      =2000, num_particulas=50,
  dimensions=3, bounds=(0, 1.5), max_iters=50, F=0.2, CR=0.5)  # 20 10
          30  20
  # Crear la carpeta DATOS si no existe
  carpeta_resultados ="DE_50_50_0.2_0.5_test10" #"DE_100_50_0.2_0.5
      _test7"
  os.makedirs(f"{carpeta_resultados}", exist_ok=True)
  best_pos, best_puntaje = system.run_DE()
```

Anexo 4. Mediciones de las longitudes de las barras de soporte

Figura 4.37
Largo de la primera barra con inicio en 20

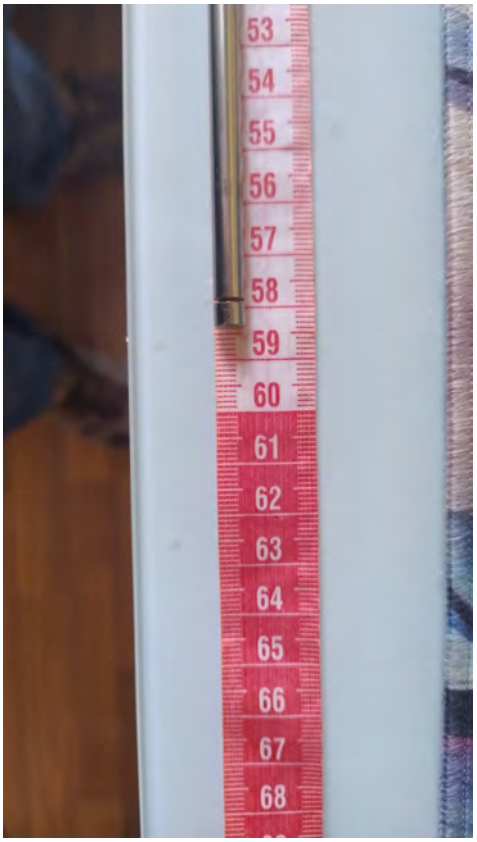
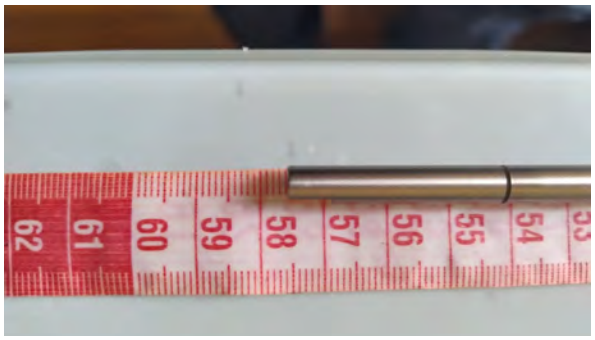


Figura 4.38
Largo de la segunda barra con inicio en 20



Anexo 5. Mediciones de las masas de los componentes impresos y masas estándar para la carga

Figura 4.39
Peso de la primera masa estándar

Figura 4.40
Peso de la segunda masa estándar

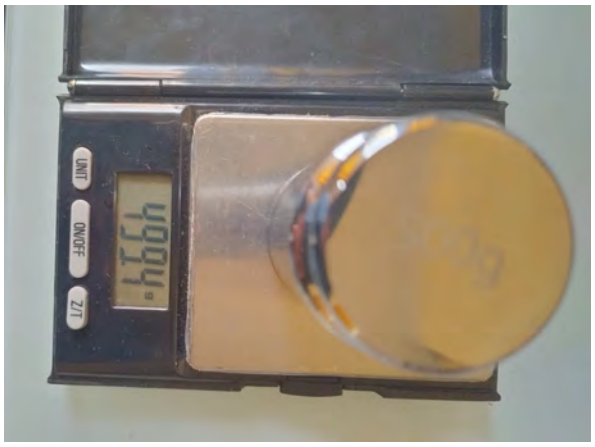


Figura 4.41
Peso de la primera barra

Figura 4.42
Peso de la segunda barra

Figura 4.43
Peso de soporte de masa base

Figura 4.44
Peso de soporte central

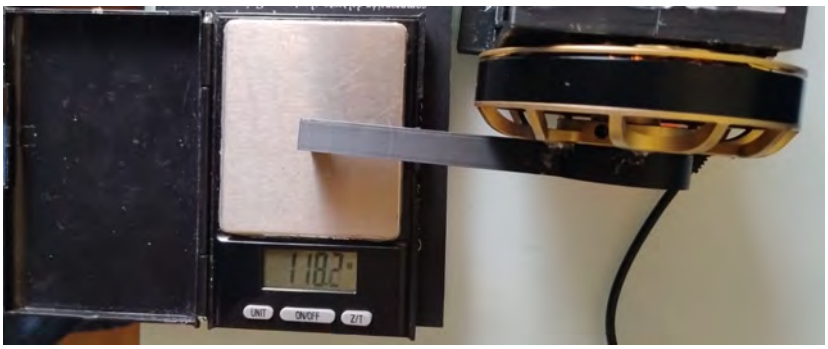
Anexo 6. Medición de la relación corriente momento inercia

Figura 4.45

Preparación para medida de la fuerza por unidad de corriente

Figura 4.46

Medición de fuerza ejercida sobre la balanza de 118 g



Anexo 5. Sistema real armado completo

Figura 4.47
Sistema real plano diagonal

Figura 4.48
Sistema real plano horizontal