ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

SIMULACIÓN DE CONTROLADORES DE SEGUIMIENTO DE TRAYECTORIA BASADOS EN INTELIGENCIA ARTIFICIAL PARA UN ROBOT HUMANOIDE NAO

SIMULACIÓN DE UN CONTROL BASADO EN ALGORITMOS GENÉTICOS PARA EL SEGUIMIENTO DE TRAYECTORIAS DE UN ROBOT HUMANOIDE

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

SEBASTIÁN GABRIEL GALARZA PÉREZ

sebastian.galarza@epn.edu.ec

DIRECTOR: DR. GEOVANNY DANILO CHÁVEZ GARCÍA

danilo.chavez@epn.edu.ec

DMQ, octubre 2023

CERTIFICACIONES

Yo, SEBASTIÁN GABRIEL GALARZA PÉREZ declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

SEBASTIÁN GABRIEL GALARZA PÉREZ

Certifico que el presente trabajo de integración curricular fue desarrollado por SEBASTIÁN GABRIEL GALARZA PÉREZ, bajo mi supervisión.

_____

DR. GEOVANNY DANILO CHÁVEZ GARCÍA

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

SEBASTIÁN GABRIEL GALARZA PÉREZ

DR. GEOVANNY DANILO CHÁVEZ GARCÍA

DEDICATORIA

El presente trabajo está dedicado a mi familia (Inés, Gerardo, Leo, Fernando, Camila, Alex, Thiago, Karina y Karen), quienes han sido un pilar fundamental en mi vida y me han brindado apoyo y motivación en todo momento.

Además, a todas aquellas personas que buscan innovar y crear nuevas formas de solucionar problemas mediante el uso de la tecnología. Espero que este trabajo contribuya como una guía para la realización de proyectos futuros.

AGRADECIMIENTO

A Dios, por guiarme e iluminarme durante todo este camino.

A mi familia, que me ha dado su apoyo y cariño incondicional.

A mis amigos Darwin, Esteban, Elvis, Joel, Karol, Luis y Renato, quienes me han impulsado a seguir adelante para cumplir mis metas y me han ayudado a sobrellevar las dificultades que se presentaron durante mi vida universitaria.

A aquellos profesores que supieron transmitir sus conocimientos e incentivar el amor hacia la carrera.

Al Dr. Danilo Chávez gracias al cual he podido realizar el presente trabajo de titulación y quien me ha guiado para la realización de este documento.

ÍNDICE DE CONTENIDO

RESUMEN

En el presente trabajo se muestran los resultados de la aplicación de una técnica de inteligencia artificial, basada en computación evolutiva, para realizar el control de seguimiento de trayectorias de un robot humanoide. La técnica mencionada se trata de algoritmos genéticos, mismos que emplean principios biológicos y conceptos obtenidos de la teoría de la evolución para resolver problemas de optimización.

Para el caso de estudio, se propone el desarrollo de un controlador para el seguimiento de trayectorias basado en Lyapunov, mismo que se diseña a partir del modelo cinemático de un robot móvil. Inicialmente, se realiza la sintonización manual del controlador, únicamente con fines comparativos. Tras obtener el controlador inicial, se procede a aplicar el algoritmo genético, de tal manera que se calculen valores optimizados para las constantes del controlador. Este algoritmo realiza una simulación iterativa offline del proceso en lazo cerrado para poder obtener los mejores valores que minimicen una función objetivo determinada por el desarrollador del presente trabajo. Los resultados obtenidos son evaluados mediante el cálculo de índices de desempeño y de la función objetivo basada en los mismos.

Posterior al cálculo de las ganancias optimizadas para el controlador, se obtienen los equivalentes de dichos valores, de tal manera que puedan ser aplicados para el control del robot humanoide NAO. El funcionamiento de dicho controlador, así como su análisis y comparación con un controlador sintonizado manualmente, serán mostrados en el presente documento; evidenciando que el trabajo desarrollado ofrece una mejoría en lo que respecta al rendimiento del sistema.

PALABRAS CLAVE: Algoritmos genéticos, inteligencia artificial, robot humanoide, control, seguimiento de trayectoria, Lyapunov.

ABSTRACT

This document shows the results of the application of an artificial intelligence technique, based on evolutionary computation, to perform the trajectory tracking control of a humanoid robot. The technique mentioned previously is based on genetic algorithms, which use biological principles and concepts obtained from evolutionary theory to solve optimization problems.

For the case of study, it is proposed the development of a controller for trajectory tracking based on Lyapunov, which is designed by using the kinematic model of a mobile robot. Initially, the controller is manually tuned for comparative purposes only. After obtaining the initial controller, the genetic algorithm is applied to calculate optimized values for the controller's constants. This algorithm performs an offline iterative simulation of the closed-loop process to obtain the best values that minimize an objective function determined by the developer of this work. The obtained results are evaluated by calculating performance indices and the objective function based on them.

After the calculation of the optimized gains for the controller, the equivalents of these values are obtained, so that they can be applied to perform the control of the NAO humanoid robot. The operation of this controller, as well as its analysis and comparison with a manually tuned controller, are presented in this document; demonstrating that the developed work offers an improvement in terms of system performance.

KEYWORDS: Genetic algorithms, artificial intelligence, humanoid robot, control, trajectory tracking, Lyapunov.

1 INTRODUCCIÓN

En la actualidad, la inteligencia artificial (IA) es uno de los campos de las ciencias computacionales que ha atraído gran interés debido, principalmente, a la diversidad de aplicaciones en las que puede ser empleada [1]. Desde su creación en la década de 1950 y hasta nuestros días, esta tecnología ha tenido un enorme desarrollo, lo cual puede evidenciarse en campos como las finanzas, la industria, la domótica, la medicina, la docencia o el transporte [2]; donde la IA ha contribuido con notables aplicaciones que facilitan el cumplimiento de ciertas tareas dentro de dichos campos. De forma general, la inteligencia artificial permite que las computadoras empleen una gran cantidad de funciones como la visión, comprensión, manejo del lenguaje, análisis de datos [3][4], así como otras características que aproximan su comportamiento con el de un ser humano y que ayudan a solucionar problemas determinados [4], incorporando las ventajas que ofrecen las máquinas como la velocidad y capacidad de procesamiento. En base a lo mencionado, y tomando en cuenta la importancia que está adquiriendo un área como la robótica en diversos ámbitos de la industria y de la vida cotidiana, resulta imprescindible analizar las posibles aplicaciones que pueden desarrollarse al integrar la inteligencia artificial en dicho campo.

La robótica es una ciencia multidisciplinaria que incorpora varias tecnologías y que permite realizar el diseño de máquinas que, mediante programación, pueden realizar tareas de forma autómata o emular el comportamiento de animales o humanos [5][6]. Con respecto a la clasificación de robots, existen varios tipos entre los que pueden mencionarse los robots industriales, móviles, humanoides o zoomórficos [6].

Los robots móviles son sistemas electromecánicos dotados de un sistema de locomoción que les permite desplazarse de forma autónoma sin estar sujetos físicamente a un punto. Además, poseen sensores que les permiten monitorear su posición respecto a su origen y su destino [7]. Estos robots tienen una amplia gama de tareas entre las que se puede mencionar: exploración planetaria o minera, misiones de búsqueda y rescate, asistencia médica, exploración marítima o ejecución de tareas rutinarias en entornos, como es el caso del seguimiento de trayectorias o caminos [7]. Los robots humanoides por su parte son un sistema robotizado, diseñado principalmente para reproducir la constitución y comportamiento de un ser humano [8]. A pesar de que su principal aplicación se encuentra en el campo de la investigación, estos robots pueden realizar actividades de asistencia, médicas, ensamblado, manipulación de objetos, etc. [9]. Esto evidencia una gran similitud entre las aplicaciones que puede realizar un robot móvil y un humanoide, teniendo en

consideración que este último posee ventajas en lo que respecta a interacción, manipulación de objetos y mejor adaptabilidad a los entornos de funcionamiento.

Para que un robot humanoide pueda cumplir de forma apropiada su tarea, es necesario contar con un modelo adecuado del robot, así como un controlador que ofrezca un buen rendimiento y robustez frente a la tarea a realizar. No obstante, la obtención de un modelo matemático para un robot humanoide suele ser un desafío debido a la complejidad inherente que presenta el sistema. Por otro lado, en lo que respecta al controlador, actualmente existen aplicaciones de inteligencia artificial como es el caso de los algoritmos genéticos, que permiten optimizar parámetros de un controlador, de tal manera que se obtenga un mejor rendimiento en el sistema. Esto se realiza mediante mecanismos de cruce, mutación y selección iterativa de las ganancias, hasta cumplir con una condición de paro o minimización de una función objetivo.

En el presente trabajo, se considera el desarrollo de un controlador para el seguimiento de trayectorias basado en Lyapunov, optimizado mediante el uso de algoritmos genéticos, de tal manera que se obtenga un controlador con un rendimiento óptimo y que permita a un robot humanoide realizar el seguimiento de trayectorias planteadas. El algoritmo genético está configurado de tal manera que, mediante una generación aleatoria y uniforme de la población, un mecanismo de cruce aritmético, un mecanismo de selección por ruleta y una técnica de mutación adaptativa factible, se pueda minimizar una función objetivo definida por el desarrollador y que permita reducir el valor de los índices de desempeño calculados para el sistema. Cabe resaltar que para el diseño del controlador, se considera el uso del modelo matemático de un robot móvil. Posteriormente, los resultados obtenidos son adaptados para el funcionamiento del robot humanoide. Finalmente, la verificación del mejoramiento en el rendimiento del sistema será analizada al comparar los resultados obtenidos, con aquellos alcanzados al emplear un controlador sintonizado de forma manual.

1.1 OBJETIVO GENERAL

Diseñar y simular un controlador basado en Algoritmos Genéticos para el seguimiento de trayectorias de un robot humanoide NAO V6.

1.2 OBJETIVOS ESPECÍFICOS

1. Realizar un estudio bibliográfico sobre la principal información del robot NAO V6, del control para el seguimiento de trayectorias de robots móviles, así como de Algoritmos Genéticos y su aplicación en sistemas de control de robots.
2. Diseñar un controlador basado en Algoritmos Genéticos, enfocándose en la optimización de un controlador basado en Lyapunov.
3. Establecer la conexión entre el software Matlab y CoppeliaSim Edu con el objetivo de realizar el envío y recepción de datos, de tal manera que el robot NAO V6 pueda realizar el respectivo movimiento sobre las trayectorias definidas.
4. Probar el funcionamiento del controlador desarrollado, verificando que el robot NAO V6 cumpla con el seguimiento de trayectorias.
5. Analizar el desempeño del controlador implementado mediante la obtención de índices de desempeño empleados en sistemas de control.

1.3 ALCANCE

- Se realizará una revisión bibliográfica sobre el robot NAO V6, sus características generales, especificaciones, estructura, etc.; así como del control de seguimiento de trayectoria para un robot móvil uniciclo. Además, se realizará un estudio sobre Algoritmos Genéticos, sus características y su aplicación para realizar la optimización de parámetros de un controlador para el seguimiento de trayectorias.
- Se diseñará un controlador basado en Lyapunov para el control de seguimiento de trayectorias de un robot móvil y se definirán las trayectorias que deberá seguir el robot. Para dicho controlador, se realizará la sintonización manual de parámetros.
- Se realizará una revisión bibliográfica sobre los índices de desempeño empleados en sistemas de control y se seleccionarán aquellos que sean idóneos para realizar la optimización de parámetros del controlador para el seguimiento de trayectorias. Posteriormente, se realizará el cálculo de dichos índices dentro del sistema de control implementado en Matlab/Simulink.
- Se realizará la optimización de parámetros del controlador basado en Lyapunov mediante el uso del toolbox de Matlab “Optimize Live Editor”, mismo que permite la

implementación de un algoritmo genético que permite minimizar una función objetivo definida por el desarrollador.

- Se ejecutarán simulaciones a través del uso de Matlab/Simulink y CoppeliaSim Edu con el objetivo de validar el funcionamiento del controlador optimizado mediante Algoritmos Genéticos para el control de seguimiento de trayectoria del robot NAO V6. En este caso, la operación del sistema se evaluará mediante la obtención de las señales de error, señales de control y la visualización 3D del movimiento del humanoide.
- Se obtendrán los índices de desempeño del sistema y se realizará un análisis de dichos valores con el objetivo de determinar el desempeño del controlador.
- El presente proyecto no contemplará la implementación física del controlador desarrollado, en el robot humanoide NAO V6.

1.4 MARCO TEÓRICO

1.4.1 ROBÓTICA

Durante las últimas décadas, la robótica ha sido uno de los campos con mayor desarrollo y se ha vuelto tendencia debido a sus avances en cuanto a aplicabilidad, seguridad y precisión para realizar tareas [10]. Cuando se hablaba de robots, existía un enfoque dirigido exclusivamente a los robots industriales, empleados para manipular piezas, materias o herramientas disponibles en la industria [11]. Sin embargo, debido a los avances tecnológicos existentes, actualmente se puede hablar de robots disponibles para varios ámbitos y que poseen formas distintas, de tal manera que puedan cumplir una amplia gama de tareas. Oficialmente, y tras varias revisiones por parte de la Organización Internacional de Estandarización (ISO), se obtiene la siguiente definición de un robot: “Se trata de un mecanismo accionado, programado con cierto grado de autonomía, que se mueve dentro de su entorno, para realizar tareas previstas” [10].

Existe una gran diversidad de criterios según los cuales pueden ser clasificados los robots, estos pueden considerar el ambiente en donde son operados, su campo de aplicación, las tareas que realizan, su locomoción o su arquitectura [12]. Una clasificación general de los robots puede considerar los siguientes tipos: industriales, móviles, humanoides, educativos y zoomórficos [6][13].

1.4.1.1 Robots Móviles

Un robot móvil es un sistema electromecánico capaz de desplazarse de un lugar a otro, sin estar sujeto a un punto. Estos poseen un sistema de sensores que les permiten monitorear su posición con respecto a un punto inicial y su lugar de destino [7]. Pueden ser controlados de forma remota con la ayuda de un operador, de forma semi autónoma, o de manera totalmente autónoma mediante controladores implementados en el sistema [12]. Existe una gran variedad de aplicaciones en las que pueden ser empleados, entre las que se puede mencionar: transporte de material, navegación por terrenos desconocidos, acceso a lugares peligrosos, distantes o inaccesibles; exploración planetaria, submarina o minera; asistencia personal, entretenimiento, asistencia médica y ejecución de tareas rutinarias como en el ámbito agrícola [6][7].

Los robots móviles pueden ser clasificados según tres sistemas de locomoción: patas, orugas y ruedas; siendo esta última la que ha tenido un mayor desarrollo debido a sus ventajas frente al resto de sistemas [7][14]. Los robots móviles con ruedas (RMR) pueden emplear cuatro tipos de ruedas para su movimiento, así como configuraciones mixtas de las mismas. Estos tipos son: convencionales, omnidireccionales, tipo castor y ruedas de bola [14].

1.4.1.2 Robots Humanoides

Los robots humanoides son sistemas electromecánicos cuya morfología y capacidades están inspiradas fundamentalmente en un ser humano, de tal manera que puedan imitar el comportamiento, percepción y cinemática de un cuerpo humano [15]. Debido a su aspecto antropomorfo, estos robots fueron concebidos desde un inicio para desempeñarse en entornos humanos, es decir, ambientes no estructurados y dinámicos a los que deben ser capaces de reaccionar para poder mantener su equilibrio [16]. Comúnmente, los humanoides suelen contar con un torso, dos brazos, dos piernas y una cabeza; sin embargo, existen algunos modelos que únicamente emulan la parte superior del torso de una persona, contando con una base móvil para su movimiento [17]. Hay una gran variedad de aplicaciones y áreas en las que pueden ser empleados, entre las que se encuentran: el campo de la investigación (mecánicas de locomoción o interacciones humano-máquina) [12], entretenimiento, asistencia, medicina, operaciones de rescate y ensamblaje, educación, etc. [9].

En la mayor parte de tareas en las que son empleados este tipo de robots, es necesario realizar desplazamientos sobre el plano x-y, por lo que la locomoción bípeda del humanoide

se convierte en uno de los principales enfoques debido a la necesidad de mantener el balance y equilibrio del robot. Ambas condiciones pueden ser solventadas si se garantiza que el punto de momento cero (ZMP) se mantenga dentro del polígono formado por los puntos de apoyo de sus pies con el suelo [18]. El ZMP es aquel punto donde el momento debido a la inercia y gravedad son cero en el eje horizontal [19]. Sin embargo, esto se vuelve un problema complejo debido a la dinámica del sistema, perturbaciones o limitaciones.

Dado que estos robots son sistemas no lineales con dinámicas acopladas y complejas, además de poseer varios grados de libertad (GDL) [16], se vuelve una tarea altamente compleja el hecho de obtener un modelo matemático que represente de forma adecuada al sistema. Por tal motivo, existe la posibilidad de emplear métodos que permitan simplificar los modelos o considerar el uso de modelos alternativos que permitan simular el sistema del robot. En este último caso, los más representativos son el modelo de péndulo invertido lineal (PIL) o modelos carro-mesa donde la masa total del robot se representa por un carro que se desplaza sobre una mesa de masa despreciable ubicada en su torso [16]. Además, cuando se trabaja con aplicaciones donde el humanoide deba realizar movimientos de caminata sobre una superficie plana, donde la orientación del robot en el plano sagital permanece tangente a la trayectoria de movimiento, es posible considerar el uso del modelo cinemático de un robot móvil uniciclo como una alternativa válida para representar el comportamiento del sistema de un humanoide [20][21][22].

1.4.2 ROBOT HUMANOIDE NAO V6

El NAO V6 es un robot humanoide autónomo y totalmente programable creado por la compañía Aldebaran Robotics [23]. Este robot puede realizar funciones de reconocimiento facial, escuchar, interactuar y hablar con los usuarios. Además, cuenta con distintos sensores, motores y articulaciones para poder caminar, bailar, o realizar distintas actividades de acuerdo con la programación establecida. Entre los principales usos establecidos para el robot NAO se encuentran la educación, investigación, robótica social, navegación y la teleoperación [15][24]. En la Figura 1.1 se puede apreciar una vista general del robot humanoide NAO, utilizado para el presente proyecto.

Figura 1.1. Robot Humanoide NAO V6 [25]

1.4.2.1 Especificaciones Técnicas

El robot NAO V6 tiene un peso de 5.48 kg, así como una altura de 57.4 cm. Este cuenta con 25 grados de libertad, distribuidos de la siguiente manera: 2 GDL en la cabeza, 5 GDL en cada uno de sus brazos, 1 GDL en el torso, 5 GDL en cada pierna y 1 GDL en cada una de sus manos [26]. En la Figura 1.2 se puede observar de mejor manera la cadena cinemática y articulaciones del robot. En cuanto a sus características informáticas, trabaja con un procesador ATOM E3845 de 1.91 GHz, 4 GB de RAM, 32 GB de almacenamiento y una memoria caché de 2 MB [26].

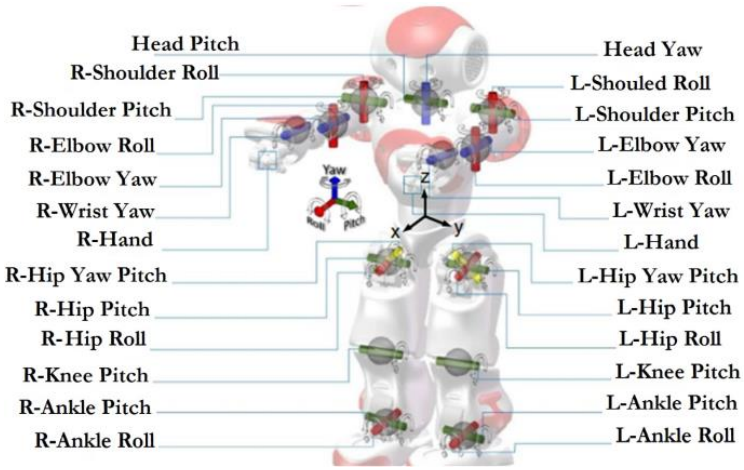

Figura 1.2. Articulaciones y cadena cinemática NAO V6 [27]

El robot dispone de cámaras, sensores táctiles, micrófonos, altavoces, manos prensiles, ojos led, sensores de proximidad, encoders, acelerómetros, giroscopios, entre otros sensores y dispositivos que le permiten cumplir sus funciones programadas, así como tener una mejor interacción con el usuario. Además, cabe mencionar que tiene la posibilidad de emplear reconocimiento de voz a elección del usuario entre 20 idiomas disponibles [26].

Los sensores disponibles en el robot realizan una actualización de datos adquiridos en ciclos de 20 ms. Cuenta también con dos girómetros y tres acelerómetros que le permiten realizar mediciones en tiempo real. Además, dispone de encoders rotativos magnéticos (MRE) que le permiten medir las posiciones reales de las articulaciones [28].

Para su autonomía, el humanoide cuenta con una batería de ion de litio de 21.6 voltios y 2.1 amperios, lo cual le otorga un tiempo de funcionamiento de 60 minutos en uso activo o 90 minutos en uso normal. También, cuenta con conexiones WiFi (IEEE 802.11) o Ethernet (RJ45) para su comunicación con computadores remotos [26].

Este humanoide cuenta con módulos de software integrados como es el caso de Choregraphe, mismo que le permite realizar la conversión de texto a voz, la localización de sonidos, detección de patrones visuales, formas y colores; así como la detección de obstáculos. Dispone además de una interfaz GUI denominada Choregraphe mediante la cual el usuario puede controlar de forma simple el movimiento del robot [28].

Su sistema operativo se denomina NAOqi y puede ser empleado en Windows, Mac OS o Linux. Además, para realizar la programación de funcionamiento del robot NAO se puede emplear un sistema de bloques de funciones, así como lenguajes de programación avanzados como Python, C++ o Java [26].

1.4.3 INTELIGENCIA ARTIFICIAL

La inteligencia artificial (IA) es un campo que no ha tenido una definición universalmente aceptada a lo largo de los últimos años debido a su constante evolución, a las distintas interpretaciones que se le han dado, así como la diversidad de técnicas y aplicaciones que se engloban dentro de esta área. A pesar de lo mencionado, se puede contemplar la siguiente definición que toma en cuenta los enfoques principales de la IA: La inteligencia artificial es la facultad que tienen las máquinas de poder aprender de un conjunto de datos, emplear algoritmos y otras funciones, así como emplear el conocimiento adquirido durante la toma de decisiones; de tal manera que su comportamiento se asemeje al de un ser humano [29]. Cabe destacar que las máquinas ofrecen ventajas significativas en lo referente a su capacidad de analizar grandes volúmenes de datos, reducir el porcentaje de error al realizar sus tareas y poder realizar sus funciones sin la necesidad de un descanso [29].

Debido a la importancia que ha adquirido en los últimos años, la inteligencia artificial ha sido considerada para su aplicación en áreas como la medicina, ingeniería, economía,

biología, robótica, física, matemáticas o informática [4]. En estos campos, la IA es usada para realizar tareas como la optimización de procesos, diagnóstico de fallas, toma de decisiones, análisis financiero y de datos, reconocimiento y análisis de imágenes, mantenimiento predictivo, lingüística computacional, entre otras [30]. De acuerdo con la tarea que se desee realizar, se pueden aplicar varias técnicas de inteligencia artificial, mismas que se basan en los fundamentos desarrollados en las distintas ramas de la IA.

Entre las ramas principales de la inteligencia artificial puede mencionarse: el aprendizaje automático (Machine Learning), la visión por computadora, el procesamiento de lenguaje natural (NLP), sistemas expertos, el aprendizaje profundo (Deep Learning), minería de datos, razonamiento y planificación, y la computación evolutiva [29][31]. Actualmente, existe una gran variedad de técnicas de IA, sin embargo, algunas de ellas han destacado por su efectividad, versatilidad y resultados obtenidos tras su aplicación. Entre estas técnicas se puede mencionar las redes neuronales artificiales, la lógica difusa, los algoritmos de agrupamiento (Clustering) y los algoritmos genéticos [4].

1.4.4 SISTEMAS DE CONTROL

Un sistema de control es aquel medio a través del cual cualquier variable o magnitud de interés para el funcionamiento de una máquina, mecanismo u otro equipo, puede ser modificada o mantenida de acuerdo con una forma deseada [32]. La principal finalidad de estos sistemas es obtener una salida deseada, con un desempeño deseado, dada una entrada específica; mediante el funcionamiento de los elementos que lo conforman [33]. Entre los principales objetivos de control para robots, se encuentran: el control de posición, el control de seguimiento de trayectorias y el control robusto.

1.4.4.1 Sistemas de control en lazo abierto

Un sistema en lazo abierto es aquel que no cuenta con una realimentación de la señal de salida y su respuesta depende directamente de la señal de entrada [32]. Estos sistemas no pueden corregir de forma automática las perturbaciones o variaciones en su salida [33]. Debido a que se caracterizan por su simplicidad e imprecisión, son usualmente aplicados para procesos no críticos.

1.4.4.2 Sistemas de control en lazo cerrado

Un sistema de control en lazo cerrado realiza una medición de la salida del sistema (conocida como señal de realimentación) y la compara con la señal de entrada, de modo que se obtiene una diferencia entre ellas (denominada señal de error) [34]. A partir de esta diferencia, el sistema de control realiza las modificaciones necesarias para que se pueda alcanzar la referencia deseada y se corrija el error [35]. En comparación con los sistemas en lazo abierto, esta configuración ofrece mayor exactitud, precisión, adaptabilidad y una menor sensibilidad a ruidos o perturbaciones [33].

1.4.4.3 Control de seguimiento de trayectorias

El control de seguimiento de trayectorias es una de las aplicaciones en las que comúnmente se emplean los robots móviles debido a que mediante esta aplicación, pueden analizarse métodos de navegación de dichos robots. Durante su desplazamiento, estos robots se encuentran cambiando constantemente su posición y orientación por lo que es necesario contar con un reconocimiento permanente del entorno mediante el uso de sensores, especialmente si se trata de un entorno desconocido [36]. De esta manera, se puede asegurar que el robot pueda adaptarse y realizar los movimientos respectivos para seguir una trayectoria o evadir obstáculos que se presenten. No obstante, en entornos conocidos, el uso de sensores es un tema secundario ya que ahora el objetivo principal se enfoca en una adecuada planificación de la trayectoria y la verificación de que el robot pueda cumplir de forma adecuada con la misma [37]. El diseño de controladores que permitan cumplir con estas funciones suele realizarse en base a los modelos cinemáticos o dinámicos del robot [38].

Con respecto a los robots humanoides, el seguimiento de trayectorias también es un enfoque importante, debido a que la mayor parte de aplicaciones en las que se emplean estos robots requieren un análisis de la locomoción bípeda de los mismos. Para que estos robots puedan realizar una caminata adecuada, se debe tomar en cuenta el equilibrio y balance del robot, lo cual lo vuelve un proceso complejo. Además, se debe buscar la manera de controlar cada una de sus articulaciones para que el humanoide cumpla con las funciones establecidas [39]. En el caso del robot NAO, se puede considerar un control de alto nivel. Esto debido a que su cinemática inversa, directa y dinámica; el equilibrio y balance durante la caminata, así como funciones de movimiento y operación de sus articulaciones, se toman en cuenta dentro de las librerías de su software NAOqi [40].

Los robots humanoides poseen una compleja estructura cinemática, misma que en muchas ocasiones provoca un incremento en las imprecisiones mecánicas o interacciones no ideales de los pies con el suelo [41]. Por lo mencionado, se presenta un problema adicional cuando se habla del seguimiento de trayectorias para robots humanoides. Esto hace referencia al movimiento oscilatorio que realiza el humanoide durante su caminata, mismo que puede ser considerado como un error y por tanto, afectar el desplazamiento y precisión del robot durante el seguimiento de una trayectoria [41].

1.4.5 DISEÑO DE CONTROLADORES EN SISTEMAS DE CONTROL

El diseño de controladores se realiza a partir de un modelo adecuado del proceso, así como una serie de especificaciones que permiten describir el comportamiento deseado del sistema. Dichas especificaciones son exclusivas para cada aplicación en particular y pueden estar enfocadas en la estabilidad del sistema, la precisión en estado estable, parámetros de la respuesta transitoria, robustez o rechazo a perturbaciones [34]. El cumplimiento de las características deseadas depende tanto de la estructura del sistema de control, así como de los parámetros del controlador [35]. Cabe mencionar que para la sintonización de los parámetros mencionados se pueden usar métodos empíricos basados en la experimentación, fórmulas o métodos basados en herramientas computacionales. Existe una gran variedad de técnicas de control disponibles que pueden ser implementadas, algunas de las cuales se muestran a continuación:

- Técnicas de Control Convencionales: son técnicas usadas principalmente cuando el objetivo de control es invariante en el tiempo y siguen los principios clásicos de control. En esta clasificación pueden encontrarse los controladores P, PD, PI, PID, redes de atraso o adelanto, entre otros [42].
- Técnicas Avanzadas de Control: se emplean en procesos donde se presentan no linealidades, perturbaciones frecuentes, requisitos de seguridad/rendimiento; y en general en donde los controladores clásicos no presentan un desempeño adecuado. Ejemplos de estas técnicas son el control en cascada, por relación, selectivo, por acción precalculada, etc. [43].
- Sistemas de Control Inteligente: son métodos de control que emplean técnicas de inteligencia artificial para optimizar, aprender, adaptarse y mejorar el rendimiento del controlador. Son usados principalmente cuando el proceso cuenta con comportamientos no lineales, existe incertidumbre en el entorno, no se cuenta con un modelo preciso del proceso o cuando se desea optimizar el funcionamiento del

sistema. Algunas de las técnicas empleadas en este caso son las redes neuronales, lógica difusa, algoritmos genéticos, sistemas expertos, entre otros [44], [45].

1.4.6 ÍNDICES DE DESEMPEÑO

Los índices de desempeño son indicadores cuantitativos que permiten determinar qué tan bueno es el rendimiento de un sistema de control en lazo cerrado. Cuando los parámetros de un controlador se ajustan de tal manera que los índices de desempeño obtengan valores mínimos (o máximos), se dice que el sistema de control es óptimo [34].

1.4.6.1 Integral del error cuadrático (ISE)

Este criterio se caracteriza por dar una mayor ponderación a los errores grandes que a los errores pequeños. Cuando el sistema de control se configura para minimizar el índice ISE, este es capaz de eliminar de forma rápida los errores grandes y tolerar errores pequeños que se mantienen por un intervalo de tiempo [46].

$$ISE = \int_0^\infty e(t)^2 \, dt \tag{1.1}$$

1.4.6.2 Integral de la salida de control cuadrática (ISU)

Este indicador permite determinar el esfuerzo que realiza el controlador durante el funcionamiento del sistema. Cuando el objetivo del sistema de control es minimizar el índice ISU, lo que se busca es reducir el consumo de energía del controlador [47].

$$ISU = \int_0^\infty u(t)^2 \, dt \tag{1.2}$$

1.4.6.3 Variaciones del esfuerzo de control (TVu)

Este criterio realiza una sumatoria de las variaciones en el esfuerzo de control, es decir, es la sumatoria de las diferencias entre la acción de control futura (u_{k+1}) y la presente (u_k). Mientras menor sea el valor de dicho índice, se dice que la señal de control es suave y por tanto, se evita el deterioro de los actuadores o elementos finales de control [48].

$$TVu = \sum_{k=1}^{\infty} |u_{k+1} - u_k| \tag{1.3}$$

1.4.7 SOFTWARE DE SIMULACIÓN

A continuación, se presentará una descripción de los programas empleados para realizar la simulación del control de seguimiento de trayectorias para el robot NAO V6.

1.4.7.1 Simulink

Simulink es una herramienta de MATLAB que permite modelar, simular y analizar sistemas dinámicos. Dentro de este entorno de simulación, la programación requerida se debe implementar mediante el uso de diagramas de bloques [49]. En el presente caso de estudio, Simulink es usado para el desarrollo del sistema de control para el robot NAO. El funcionamiento de dicho sistema de control requiere del envío y recepción de datos entre Simulink y un entorno de desarrollo para Python, para lo cual se emplean los bloques mostrados en la Figura 1.3, mismos que permiten establecer una comunicación UDP entre ambos programas. En este caso, se deben configurar valores como direcciones IP remotas y locales, puertos de comunicación, tamaño de datos y tiempos de muestreo.

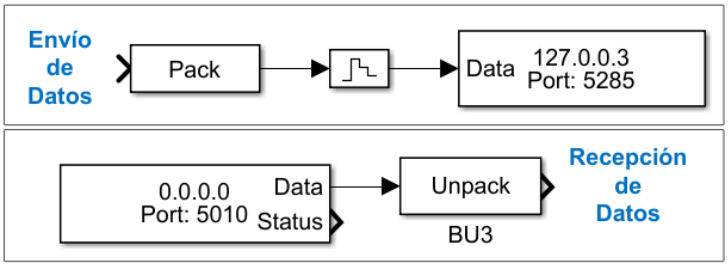

Figura 1.3. Bloques para el envío y recepción de datos por comunicación UDP.

1.4.7.2 CoppeliaSim Edu

Se trata de un simulador de robótica empleado para el desarrollo de algoritmos, creación y verificación de prototipos de robots, robótica educacional, monitoreo remoto, simulaciones de automatización de fábricas, entre otras aplicaciones. Este software dispone de un IDE que se basa en una arquitectura de control distribuido en donde cada elemento o modelo que se coloque en el entorno virtual puede ser controlado de forma individual. Dicho control se puede realizar mediante scripts integrados, plugins, nodos ROS o clientes API remotos [50]. Este último método de control permite la conexión de CoppeliaSim con programas o aplicaciones externas, como es el caso de entornos de desarrollo para Python, mediante el uso de APIs remotos [47]. Además, cabe mencionar que los controladores pueden ser programados en C/C++, Java, Lua, Python, Matlab u Octave [50]. En la Figura 1.4 se presenta el entorno virtual de simulación CoppeliaSim Edu.

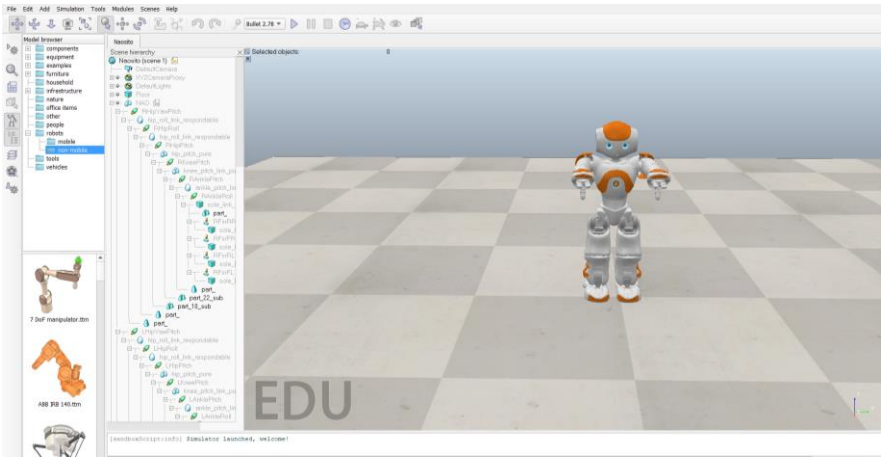

Figura 1.4. Interfaz del software CoppeliaSim Edu

1.4.7.3 Spyder

Spyder es un entorno de desarrollo de código abierto que está escrito en Python y para Python. Este cuenta con características para la edición, análisis, depuración y creación de perfiles. El editor de este software es multilenguaje y proporciona un navegador de funciones, herramientas de análisis de código y finalización automática de código [51].

1.4.7.4 NAOqi

NAOqi es el principal software del robot NAO que al ejecutarse, permite el funcionamiento del robot; además, facilita la programación, comunicación e intercambio de información entre los módulos de video, audio y movimiento del humanoide. Este software funciona como un intermediario y, al iniciarse, realiza la ejecución de las librerías necesarias para cumplir con la operación del robot [47]. NAOqi es un software que puede ser ejecutado en varios sistemas operativos como Windows, Mac OS o Linux [40].

1.4.7.5 Choregraphe

Choregraphe es una aplicación multiplataforma que permite crear animaciones o comportamientos, probarlos sobre un robot simulado o uno real, así como controlar y monitorear un robot NAO. Dentro de este software se puede crear comportamientos complejos como bailes, interacción con personas, caminatas, entre otros; sin la necesidad de escribir líneas de código [52]. Esto se debe a que el programa cuenta con bloques, mismos que ya disponen de la programación necesaria para cumplir con las acciones deseadas [47]. Una vez configurados los bloques, NAOqi se encarga de interpretarlos y

ejecutar las acciones en el robot [52]. En la Figura 1.5 se muestra la interfaz del software donde pueden ser configurados comportamientos del robot NAO.

Además de la programación en bloques, también se puede programar los comportamientos del robot usando el SDK (kit de desarrollo de software) de Python con NAOqi [47]. En ambos casos se puede obtener resultados similares, sin embargo, se debe tener en cuenta que los códigos y funciones implementados en Choregraphe se ejecutan de forma más lenta que aquellos programados en Python o C++ [52].

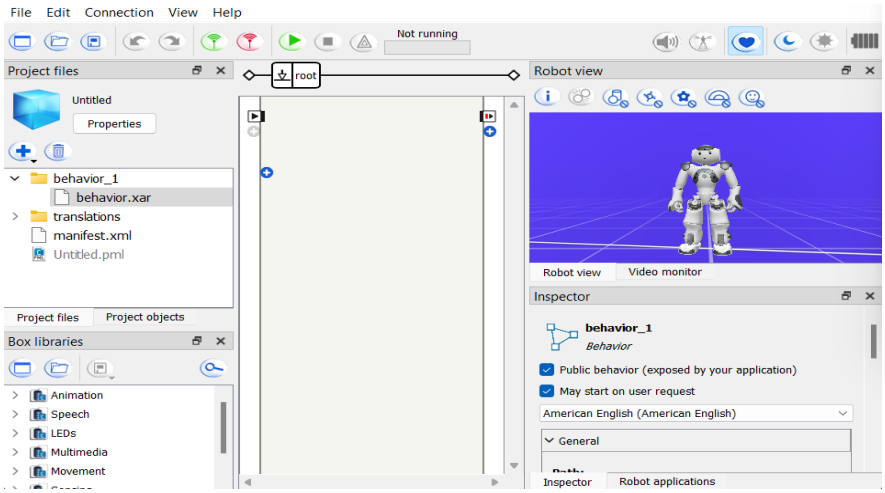

Figura 1.5. Interfaz del software Choregraphe

2 METODOLOGÍA

2.1 ESTRUCTURA DEL SISTEMA DE CONTROL

En la Figura 2.1 se presenta el esquema del sistema propuesto para que se pueda realizar el control de seguimiento de trayectorias para un robot NAO V6.

Para cumplir con el seguimiento de trayectorias planteado, se propone el desarrollo de un controlador basado en Lyapunov, cuyos parámetros serán optimizados mediante el uso de algoritmos genéticos. El algoritmo genético será configurado en el software Matlab, mediante el uso de la herramienta “Optimize Live Editor”, en donde su objetivo principal será minimizar el valor de una función objetivo calculada a partir de los índices de desempeño del sistema. Entre las características principales de este algoritmo se encuentra una generación aleatoria y uniforme de la población, un mecanismo de cruce aritmético, un mecanismo de selección por ruleta y una técnica de mutación adaptativa factible. Cada una de estas características del algoritmo serán explicadas de mejor manera en secciones posteriores. Cabe destacar que la optimización se realizará offline, mediante

el uso del modelo cinemático del robot móvil, por lo que los valores obtenidos tras la implementación del algoritmo deben ser adaptados para el control del humanoide NAO V6. El funcionamiento del controlador optimizado será evaluado mediante simulaciones, empleando los programas Matlab/Simulink y CoppeliaSim Edu, siendo este último el que permitirá visualizar el movimiento del robot en un entorno virtual. Finalmente, se podrá verificar el funcionamiento del sistema y una mejoría en su rendimiento al comparar los resultados de simulación con aquellos obtenidos mediante el uso de un controlador sintonizado manualmente.

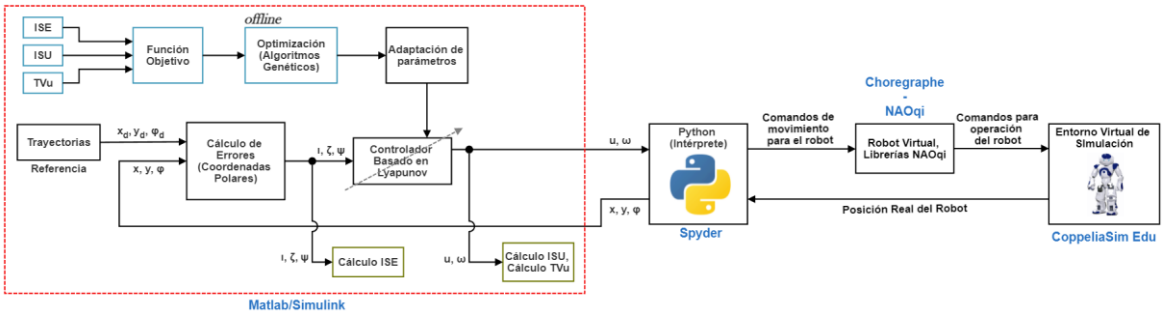

Figura 2.1. Estructura del sistema de control propuesto

Como se puede evidenciar en la Figura 2.1, en Simulink se definen las referencias de movimiento para el robot, se cuantifican los errores, se calculan las acciones de control (velocidad lineal (u) y velocidad angular (ω)), y se computan los valores de los índices de desempeño del sistema. Además, en Matlab se realiza la optimización, por algoritmos genéticos, de los parámetros del controlador basado en Lyapunov.

Para que las señales de control puedan ser enviadas desde Simulink a CoppeliaSim Edu y los valores de posición actual del robot sean enviados de CoppeliaSim Edu a Simulink, es necesario establecer la comunicación entre ambos programas. Para ello, se emplea el entorno de desarrollo de Python como un intermediario para la comunicación. Inicialmente, en Simulink se emplean los bloques mostrados en la Figura 1.3 para establecer la conexión entre este programa y Python. En este caso, los datos que se envíen o se reciban en Simulink estarán empaquetados en forma de array. Por otro lado, en CoppeliaSim Edu se configura un API remoto mediante el uso de un cuboide. A partir de esta configuración, se emplean funciones y comandos de Python que permiten obtener las posiciones en *x*, *y*, y la orientación del robot; mismas que posteriormente son enviadas a Simulink mediante comunicación UDP.

En el caso de las señales de control, el bloque UDP send de Simulink permite enviar dichos datos a Python, en donde se emplean funciones del SDK para definir los movimientos que el robot NAO deberá cumplir. Una vez programados los comandos de movimiento, estos

son interpretados por un robot virtual en Choregraphe, gracias al funcionamiento de NAOqi. Finalmente, se ejecutan las funciones de caminata y posicionamiento del robot, mismas que pueden ser visualizadas en el entorno virtual de simulación (CoppeliaSim Edu).

2.2 MODELO MATEMÁTICO DEL ROBOT

En el presente caso de estudio, y tomando en cuenta lo mencionado en la sección 1.4.1.2, como modelo matemático del robot humanoide se considera al modelo cinemático de un robot móvil uniciclo, en configuración diferencial. Esto también se sustenta en que la operación del robot se la realiza a bajas velocidades y se emplea un control en alto nivel, por lo cual, se puede asemejar su comportamiento con el de un robot móvil.

Los robots móviles en configuración diferencial se caracterizan por tener dos ruedas montadas sobre un único eje y que son controladas de forma independiente mediante motores. Además, para poder mantener el equilibrio del robot, se emplea una o dos ruedas locas de apoyo, colocadas en configuraciones triangulares o romboidales dependiendo del caso. Usualmente, estos robots son empleados para aplicaciones simples, en entornos cotidianos y con pocas exigencias [36].

En este robot móvil, sus dos ruedas proporcionan la tracción y el direccionamiento. Para efectuar movimientos en línea recta, se requiere que ambas ruedas giren en el mismo sentido y a la misma velocidad. Por otro lado, para realizar giros en cualquier dirección, la velocidad de las ruedas debe ser diferente [36].

Para que se pueda realizar el control de este tipo de robots y que estos ejecuten tareas como la navegación o el seguimiento de trayectorias, es común emplear su modelo cinemático a la hora de diseñar controladores.

2.2.1 MODELO CINEMÁTICO DEL ROBOT MÓVIL

El modelamiento cinemático de un robot se encarga de estudiar el movimiento de los sistemas mecánicos que lo conforman, sin considerar el efecto de fuerzas externas [53]. En el caso de un robot móvil de configuración diferencial, un modelo cinemático busca obtener una relación entre las velocidades del robot y las variables de posición de este.

2.2.1.1 Modelo cinemático, coordenadas cartesianas

Para poder obtener los parámetros que definen el modelo cinemático de un robot móvil diferencial, se parte de la configuración presentada en la Figura 2.2. En este caso, se puede observar un robot móvil ubicado en una posición {P}, sobre un sistema de referencia XY. Además, se define un sistema de referencia X_R-Y_R, mismo que sirve como ayuda para obtener valores de orientación y traslación del robot móvil respecto al sistema de referencia XY.

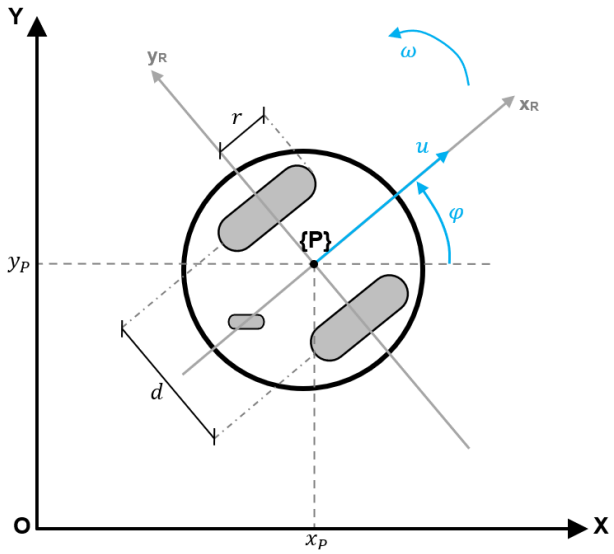

Figura 2.2. Cinemática de un robot móvil diferencial

En este contexto, se puede definir a r como el radio de las ruedas del robot, d como la distancia de separación entre cada una de las ruedas, ω y u representan la velocidad angular y lineal del robot, respectivamente; y φ es la orientación del móvil. Este último parámetro puede definirse como el ángulo descrito entre el eje X positivo y la dirección de movimiento del robot.

Las variables de estado del robot entonces son la posición en X (x), su posición en Y (y) y su orientación (φ). En base a esto, se pueden determinar las siguientes ecuaciones que representan el modelo cinemático del robot [36]:

$$\dot{x} = u \cdot \cos(\varphi) \tag{2.1}$$

$$\dot{y} = u \cdot \sin(\varphi) \tag{2.2}$$

$$\dot{\varphi} = \omega \tag{2.3}$$

Representando el modelo de forma matricial, se obtiene:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & 0 \\ \sin(\varphi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ \omega \end{bmatrix} \tag{2.4}$$

2.2.1.2 Modelo cinemático, coordenadas polares

En (2.4) se mostró un modelo para el robot móvil diferencial, sin embargo, es importante analizar las características cinemáticas de este en coordenadas polares, dado que las mismas serán empleadas para el diseño del controlador basado en Lyapunov. Por tal motivo, se parte de la configuración mostrada en la Figura 2.3**.** En este caso, se tiene el mismo robot móvil presentado en la Figura 2.2., sin embargo, se define un punto {D} que representa el objetivo de movimiento del robot (punto final de su traslación). Además, se cuenta con el sistema de referencia X_{Rd}-Y_{Rd}, mismo que permite obtener la orientación y posición deseadas, en las que se debe colocar el robot al final de su movimiento. Cabe destacar que el sistema de referencia X_{Rd}-Y_{Rd} se considera como el marco de referencia principal (global) [14].

Las coordenadas polares del robot pueden ser definidas mediante tres parámetros: ι que representa la distancia desde la posición inicial ({P}) hasta el punto final de movimiento ({D}), ζ que representa el ángulo de direccionamiento para alcanzar el punto {D} y ψ que es el ángulo de orientación con respecto al sistema de referencia del punto objetivo (X_{Rd}-Y_{Rd}).

En base a lo mencionado, el modelo cinemático del robot móvil diferencial en coordenadas polares puede ser descrito mediante las ecuaciones (2.5), (2.6) y (2.7) [14].

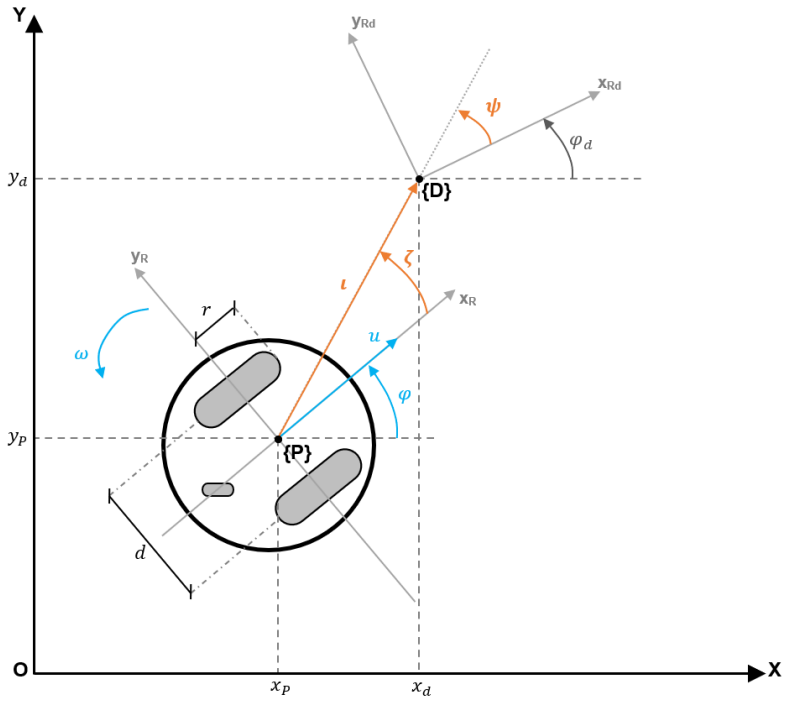

Figura 2.3. Cinemática del robot móvil diferencial, coordenadas polares

$$\dot{\iota} = -u \cdot \cos(\zeta) \tag{2.5}$$

$$\dot{\zeta} = -\omega + \left(\frac{u}{\iota}\right) \cdot \sin(\zeta) \tag{2.6}$$

$$\dot{\psi} = \left(\frac{u}{\iota}\right) \cdot \sin(\zeta) \tag{2.7}$$

En base a lo mostrado en la Figura 2.3., los parámetros ι, ζ y ψ pueden ser descritos mediante las siguientes ecuaciones:

$$\iota = \sqrt{(x_d - x)^2 + (y_d - y)^2} \tag{2.8}$$

$$\zeta = \arctan[(y_d - y), (x_d - x)] - \varphi \tag{2.9}$$

$$\psi = \arctan[(y_d - y), (x_d - x)] - \varphi_d \tag{2.10}$$

2.3 CONTROLADOR BASADO EN LYAPUNOV

Antes de realizar el diseño del controlador para el sistema, es necesario efectuar un análisis sobre el criterio de estabilidad de Lyapunov. En sistemas lineales e invariantes en el tiempo, existe una gran variedad de técnicas para determinar la estabilidad, siendo un ejemplo el criterio de Routh-Hurwitz. Sin embargo, debido a que los sistemas robóticos se distinguen por tener comportamientos no lineales y características complejas, las técnicas mencionadas no pueden ser aplicadas. Es en estos casos que se emplea el criterio de estabilidad de Lyapunov, ya que este permite determinar la estabilidad en sistemas lineales, no lineales, variantes o invariantes en el tiempo [14][54].

Existen dos métodos propuestos por Aleksandr Lyapunov para determinar la estabilidad de sistemas dinámicos, descritos por ecuaciones diferenciales:

- Primer método: Este engloba los procedimientos en donde la estabilidad puede ser determinada mediante las ecuaciones linealizadas. Es decir, donde se cuenta con la forma explícita de la solución de las ecuaciones diferenciales [54].
- Segundo método: También conocido como método directo. Este se caracteriza por no requerir la solución de las ecuaciones diferenciales [54].

El método directo de Lyapunov tiene un enfoque energético al analizar la estabilidad de un sistema. Este método toma una función definida positiva $V(x)$, misma que representa la función de energía del sistema. Si dicha función es decreciente de forma continua (posee una derivada negativa), entonces se dice que esta podrá alcanzar un estado de equilibrio

y, por tanto, tiene propiedades de estabilidad [55]. En este caso, la función definida positiva se denomina “candidata de Lyapunov”.

Debe mencionarse que, un sistema es estable si sus variables de estado permanecen acotadas dentro de un rango finito cercano al punto o estado de equilibrio. Por otro lado, un sistema es asintóticamente estable si, además de permanecer acotadas, sus variables de estado tienden a converger de forma gradual hacia el punto de equilibrio [54].

Para que un sistema cuente con un punto de equilibrio ubicado en el origen ($x = 0$) y este sea asintóticamente estable, la función candidata de Lyapunov debe cumplir con las siguientes condiciones [14]:

a. $V(x)$ y sus derivadas, existen y son continuas

b. $V(0) = 0$

c. $V(x) > 0$, si x ≠ 0

d. $\dot{V}(x) < 0$, si x ≠ 0

Para el diseño de un controlador basado en Lyapunov, se debe seguir dos reglas principales [14]:

1. Seleccionar una función candidata de Lyapunov que cumpla con las condiciones a., b. y c. mostradas anteriormente.

2. Obtener la derivada $\dot{V}(\boldsymbol{x})$, a lo largo de una trayectoria del sistema $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{r}, t)$. A partir de ello, se debe determinar una ley de control de realimentación $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{x})$; de tal manera que se cumpla que $\dot{V}(\boldsymbol{x}) < 0$ para $\boldsymbol{x} \neq 0$. Se debe tener en cuenta que $\boldsymbol{r}(\boldsymbol{x})$ es normalmente una función no lineal de x que contiene parámetros y ganancias que son definidos de forma que se cumpla la condición d. para las funciones candidatas de Lyapunov.

En el presente caso de estudio, se busca determinar señales de control que cumplan con lo siguiente:

$$\begin{bmatrix} u \\ \omega \end{bmatrix} = \boldsymbol{r}(\iota, \zeta, \psi) \tag{2.11}$$

En donde, u representa la velocidad lineal del robot y ω es su velocidad angular. Para garantizar que ι, ζ y ψ tiendan de forma asintótica a cero, se define la siguiente función candidata de Lyapunov:

$$V(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} \tag{2.12}$$

En este caso, se considera que:

$$\boldsymbol{x} = [\iota, \zeta, \psi]^T \quad ; \quad \boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & q_2 \end{bmatrix} \text{ con } q_2 > 0 \tag{2.13}$$

Reemplazando (2.13) en (2.12), se obtiene que la función candidata de Lyapunov es:

$$V(\boldsymbol{x}) = \frac{1}{2}\iota^2 + \frac{1}{2}\zeta^2 + \frac{1}{2}q_2\psi^2 \tag{2.14}$$

A partir de ello, se puede verificar que se cumple con la primera condición para funciones candidatas de Lyapunov, ya que $V(\boldsymbol{x})$ y sus derivadas son continuas, debido a que se trata de una función cuadrática. Con respecto a la segunda condición, se puede verificar su cumplimiento debido a que $V(\boldsymbol{x})$ tendrá un valor de cero siempre que $\boldsymbol{x} = 0$. Además, se observa el cumplimiento de la tercera condición, ya que (2.14) se trata de un polinomio de segundo grado, cuyo valor siempre será positivo para todo ι, ζ y ψ diferentes de cero.

Derivando la función candidata de Lyapunov respecto al tiempo, se obtiene:

$$\dot{V}(\boldsymbol{x}) = \boldsymbol{x}^T Q\dot{\boldsymbol{x}} = \iota i + \zeta\dot{\zeta} + q_2\psi\dot{\psi} = \dot{V}_1 + \dot{V}_2 \tag{2.15}$$

Para definir $\dot{V}_1$ y $\dot{V}_2$ se considera (2.5), (2.6) y (2.7), entonces se tiene:

$$\dot{V}_1 = \iota i = -\iota \cdot u \cdot \cos(\zeta) \tag{2.16}$$

$$\dot{V}_2 = \zeta\dot{\zeta} + q_2\psi\dot{\psi} = \zeta\left[-\omega + \left(\frac{u}{\iota}\right)\cdot \sin(\zeta)\right] + \left(\frac{q_2}{\iota}\right)\cdot \sin(\zeta)\cdot\psi u \tag{2.17}$$

Para cumplir la condición $\dot{V}(\boldsymbol{x}) < 0$, se debe asegurar que $\dot{V}_1 < 0$ y $\dot{V}_2 < 0$. Entonces, para obtener un $\dot{V}_1$ negativo, se considera la velocidad lineal u de la siguiente forma:

$$u = K_1 \cdot \cos(\zeta) \cdot \iota, \text{ con } K_1 > 0 \tag{2.18}$$

Reemplazando (2.18) en (2.16):

$$\dot{V}_1 = -K_1\iota^2\cos^2(\zeta) \tag{2.19}$$

Esta ecuación evidencia que $\dot{V}_1 < 0$ para todo $\boldsymbol{x} \neq 0$. Reemplazando (2.18) en (2.17) y simplificando la ecuación resultante:

$$\dot{V}_2 = \zeta \left[-\omega + \frac{K_1}{\zeta} \cdot \cos(\zeta) \cdot \sin(\zeta) \cdot (\zeta + q_2 \psi) \right] \tag{2.20}$$

Para obtener un $\dot{V}_2$ negativo, se considera la siguiente expresión para la velocidad angular ω:

$$\omega = K_2 \zeta + \frac{K_1}{\zeta} \cdot \cos(\zeta) \cdot \sin(\zeta) \cdot (\zeta + q_2 \psi), \text{ con } K_2 > 0 \tag{2.21}$$

Reemplazando (2.21) en (2.20):

$$\dot{V}_2 = -K_2 \zeta^2 \tag{2.22}$$

Esta ecuación evidencia que $\dot{V}_2 < 0$ para todo $\boldsymbol{x} \neq 0$. Entonces, considerando (2.19) y (2.22), se puede obtener el siguiente resultado con respecto a la derivada de la función candidata de Lyapunov, en donde se observa el cumplimiento de la última condición:

$$\dot{V}(\boldsymbol{x}) = -K_1 \iota^2 \cos^2(\zeta) - K_2 \zeta^2 \ < 0 \tag{2.23}$$

Por lo tanto, en (2.24) y (2.25) se puede evidenciar las ecuaciones que conforman el controlador basado en Lyapunov, tomando en cuenta el uso de la función tangente hiperbólica:

$$u = K_1 \cdot \cos(\zeta) \cdot \tanh(\iota) \tag{2.24}$$

$$\omega = K_2 \zeta + \frac{K_1}{\zeta} \cdot \cos(\zeta) \cdot \sin(\zeta) \cdot (\zeta + q_2 \psi) \cdot \frac{\tanh(\iota)}{\iota} \tag{2.25}$$

Cabe mencionar que para poder realizar el cálculo de las señales de control, primero deben ser computados los errores ι, ζ y ψ, a través de (2.8), (2.9) y (2.10).

2.4 TRAYECTORIAS PROPUESTAS

Para el presente trabajo, se consideraron tres trayectorias de referencia, mismas que pueden ser visualizadas en la Figura 2.4. Estas trayectorias son descritas en función del

tiempo de simulación y están definidas mediante ecuaciones que describen el movimiento que debe realizar el robot sobre los ejes x y y.

Los valores de referencia que se obtienen a partir de la generación de trayectorias, y que permiten realizar el cálculo de errores son: el valor deseado de posición en x (x_d), el valor deseado de posición en y (y_d), y la orientación deseada para el robot (φ_d).

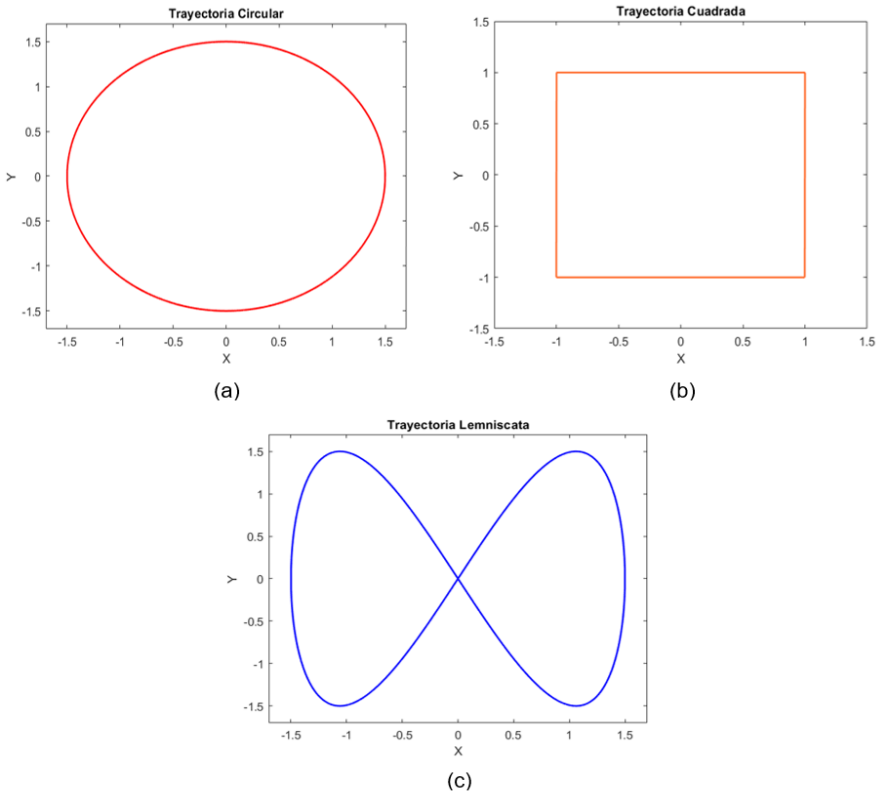

Figura 2.4. Trayectorias de referencia. (a) Círculo, (b) Cuadrado, (c) Lemniscata

En la Tabla 2.1 se presentan las ecuaciones que permiten describir las trayectorias. Cabe destacar que para poder obtener la orientación deseada del robot, se emplean las derivadas de las ecuaciones mostradas.

Tabla 2.1. Ecuaciones para Trayectorias Propuestas

Trayectoria	**EJE X (x_d)**	**EJE Y (y_d)**
Círculo	$1.5 * \cos(t/450)$	$1.5 * \sin(t/450)$
Cuadrado ($b = 400$)	$\begin{matrix} 1-(2/b)t, & t < b \\ -1, & b \le t < 2b \\ (2/b)(t-2b)-1, & 2b \le t < 3b \\ 1, & 3b \le t < 4b \\ 1-(2/b)(t-4b), & 4b \le t < 5b \\ -1 & t \ge 5b \end{matrix}$	$\begin{matrix} 1, & t < b \\ 1-(2/b)(t-b), & b \le t < 2b \\ -1, & 2b \le t < 3b \\ (2/b)(t-3b)-1, & 3b \le t < 4b \\ 1, & 4b \le t < 5b \\ 1 & t \ge 5b \end{matrix}$
Lemniscata	$1.5 * \sin(t/450)$	$1.5 * \sin(t/225)$

2.5 ALGORITMOS GENÉTICOS

La computación evolutiva (CE) es una rama de la inteligencia artificial que emplea distintas variantes de algoritmos evolutivos (AE) como una herramienta para poder trabajar con sistemas no lineales complejos. La CE cuenta con técnicas heurísticas que se basan en los principios de la evolución y que permiten resolver problemas computacionales de búsqueda y optimización. Para la resolución de dichos problemas, las técnicas de la CE adoptan una terminología similar a los conceptos biológicos para poder definir sus componentes estructurales y operaciones logarítmicas. Existen cuatro técnicas principales de la computación evolutiva: estrategias evolutivas, programación evolutiva, programación genética y algoritmos genéticos [56].

Los algoritmos genéticos (AG) son técnicas de optimización propuestas por John Holland, que basan sus mecanismos en los principios de la teoría de la evolución de Darwin para poder encontrar la solución a un problema planteado. Estos algoritmos trabajan con una población inicial formada por individuos que, por el empleo de operadores genéticos, pueden ser modificados, combinados o reemplazados en cada generación; de modo que se adapten de mejor manera al entorno (el más apto sobrevive y el menos apto se descarta) [57].

Dentro de la población inicial formada por individuos, cada uno de ellos representa una posible solución del problema a optimizar. Además, estos deben tener una determinada codificación mediante un sistema de representación. Por otro lado, debe mencionarse que cada uno de los individuos es evaluado y puntuado mediante una función de aptitud definida para el sistema [56]. De acuerdo con el nivel de adaptación o puntuación que obtenga cada individuo, estos son escogidos mediante mecanismos de selección que determinarán quienes pueden reproducirse para crear nuevas generaciones. Existe una mayor probabilidad de ser seleccionado mientras mayor sea la adaptación del individuo al entorno [58]. Cabe destacar que no todos los elegidos son aquellos que cumplen con las mejores características, ya que también se puede disponer de individuos “imperfectos” que permitan traer una mayor diversidad a la hora de realizar el cruce.

Una vez escogidos los individuos, estos mezclan sus características genéticas con otros, de tal manera que se obtenga una nueva población en donde los hijos posean mejores características que sus progenitores. Además, en esta etapa puede presentarse un proceso de mutación, en donde el valor del alelo del individuo cambia de forma aleatoria. Este proceso suele repetirse de forma iterativa, alcanzando nuevas generaciones que se aproximen cada vez más al valor óptimo deseado [43].

2.5.1 CONCEPTOS BIOLÓGICOS

Como puede evidenciarse, existen ciertos términos empleados en algoritmos genéticos que tienen sus bases en la biología. Además, en la Figura 2.5 se pueden observar ciertas expresiones utilizadas para referirse a elementos en una población. Por tal motivo, y para poder entender de mejor manera los conceptos previamente expuestos y los que se mostrarán en secciones posteriores, es necesario proporcionar los siguientes conceptos:

- **Cromosoma:** Estructura compacta y alargada, constituida principalmente por ADN. Estos contienen el material genético del organismo, además la mitad de cada cromosoma proviene de cada uno de los padres. Estos se encuentran formados por genes y, en algoritmos genéticos, se los conoce como individuos [58][59].
- **Gen:** Son la unidad básica de los cromosomas y permiten el almacenamiento de información genética. Estos llevan el material hereditario y son capaces de reproducirse o mutar. Además, están dispuestos en un orden determinado dentro del cromosoma. En algoritmos genéticos, se trata de cada bit o número que conforma un individuo [58][59].
- **Alelo:** Se trata de los valores con los que se representa un gen en un determinado locus [60].
- **Locus:** Posición invariable de un gen dentro del cromosoma [59].
- **Población:** Es el conjunto de cromosomas o individuos [60].
- **Fenotipo:** Rasgos observables o características de un individuo [58].
- **Genotipo:** Conjunto de genes presentes en la totalidad de cromosomas de un organismo [59].

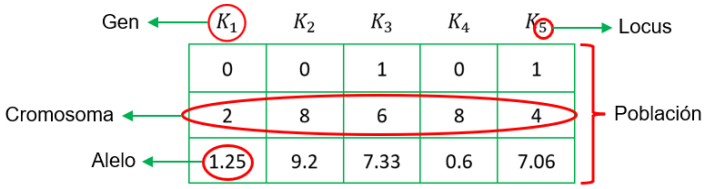

Figura 2.5. Elementos de una población

2.5.2 ESTRUCTURA DEL ALGORÍTMO GENÉTICO

Para poder implementar un algoritmo genético, se debe realizar una serie de pasos, a través de los cuales una población inicial puede ser optimizada. En la Figura 2.6 se puede evidenciar el procedimiento general que un AG realiza, mientras que en la Figura 2.7 se

muestra un diagrama de flujo en el que se evidencia el proceso que debe cumplirse para la optimización de los parámetros de un controlador para el seguimiento de trayectorias.

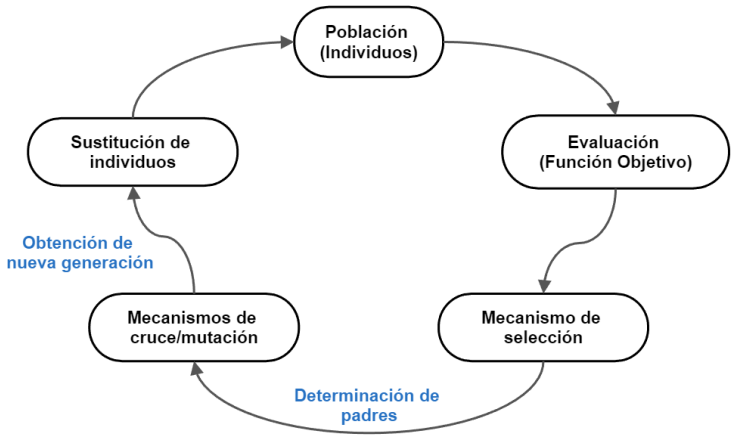

Figura 2.6. Ciclo general de un Algoritmo Genético

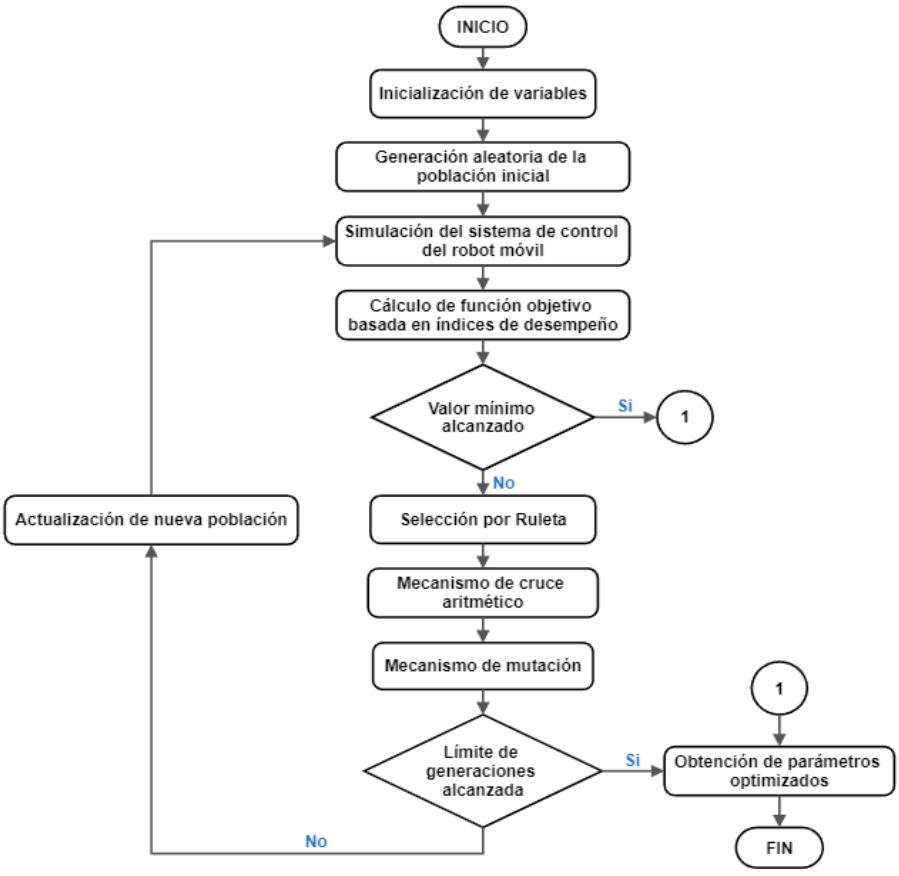

Figura 2.7. Diagrama de flujo, aplicación de AG para optimizar parámetros de controlador para el seguimiento de trayectorias

2.5.2.1 Representación de Variables

En los algoritmos genéticos, dependiendo del problema que se desee resolver, cada individuo dentro de una población debe ser representado mediante una codificación binaria

o una codificación real. En los inicios de los AG, J. Holland propuso una codificación binaria para los cromosomas, sin embargo, esta representación resulta ser poco eficiente para ciertos problemas de optimización, por lo que es necesario considerar codificaciones reales o enteras [56]. Para el presente proyecto, debido a que las variables que se desea optimizar son los parámetros de un controlador, se considera el uso de una codificación real, en donde cada individuo de la población es un vector formado por números reales. En este caso, cada gen del cromosoma representa un parámetro del controlador.

2.5.2.2 Inicialización de variables y generación de la población inicial

En las etapas iniciales del algoritmo genético, se establecen los valores de los parámetros del AG, tales como el tamaño de la población, porcentaje de cruce, límites de variables, etc.; así como otros valores necesarios para resolver el problema de optimización. Además, es necesario generar el grupo de cromosomas iniciales con los que trabajará el algoritmo. Normalmente, este proceso se realiza de forma aleatoria, sin embargo, también se puede partir de técnicas heurísticas u optimizaciones locales para acelerar la convergencia del AG [56]. En este proyecto, los cromosomas de la población inicial son generados de forma aleatoria, considerando una distribución uniforme de probabilidad; es decir, los valores de cada gen son elegidos al azar, tomando en cuenta los límites de variación de cada parámetro.

2.5.2.3 Función objetivo y condiciones de paro

La función objetivo o fitness es una expresión matemática que permite determinar el grado de adaptación de cada individuo dentro de la población a través de un valor medible. Esta medida es un indicador que define qué tan adecuadas son las soluciones para el problema [58]. En el presente caso de estudio, se considera una función objetivo basada en los índices de desempeño del sistema, específicamente los índices ISU, ISE y TVu. Cabe destacar que por la configuración del sistema, se cuenta con tres índices ISE (por los errores ι, ζ y ψ), dos índices ISU y dos índices TVu (por las señales de control u y ω).

Dicha función objetivo fue determinada de forma heurística, mediante pruebas en simulación, a partir de las cuales se consideró una expresión matemática para dicha función que asigne a cada índice de desempeño una ponderación equitativa. Esto con el objetivo de que la optimización del controlador tenga efecto sobre los errores del sistema y las señales de control. Para ello, inicialmente se obtuvo un equivalente de cada índice de desempeño, como se muestra en (2.26), (2.27) y (2.28).

$$ISE_T = \frac{0.5677 \cdot ISE_\iota + ISE_\zeta + 0.08809 \cdot ISE_\psi}{3} \quad (2.26)$$

$$ISU_T = \frac{ISU_u + 0.0264 \cdot ISU_\omega}{2} \quad (2.27)$$

$$TVu_T = \frac{TVu_u + 0.0281 \cdot TVu_\omega}{2} \quad (2.28)$$

A partir de las ecuaciones presentadas, se obtiene la siguiente expresión para la función objetivo:

$$F_{obj} = 0.655 \cdot TVu_T + 0.045 \cdot ISE_T + 0.3 \cdot ISU_T \quad (2.29)$$

Una vez que el AG se encuentra en ejecución, existen tres condiciones que pueden cumplirse para culminar con su funcionamiento. La primera es si se ha alcanzado un número máximo de generaciones definidas por el usuario, mientras que la segunda se efectúa si se determina que la función objetivo ha alcanzado un valor mínimo deseado. En cuanto a la tercera, esta se trata de un criterio de convergencia: el funcionamiento del AG culmina si tras un determinado número de iteraciones, no existe un cambio significativo en el valor de la función fitness, dentro de un rango de tolerancia.

2.5.2.4 Operadores Genéticos

Los operadores genéticos son herramientas que permiten la obtención de nuevas generaciones de individuos a partir de cromosomas padres. Existen tres operadores empleados en algoritmos genéticos: Selección, Cruce y Mutación.

2.5.2.4.1 Selección

La selección de individuos se realiza mediante procesos iterativos en donde se determinan los mejores cromosomas dentro de una población, para que puedan reproducirse y formar nuevas generaciones. La selección de individuos se realiza en base a la función objetivo, además, hay que tener en cuenta que no siempre se consideran los cromosomas con mejores características, ya que aquellos con un valor bajo en la función objetivo también pueden ser seleccionados para dar una mayor diversidad a la población y evitar convergencias prematuras del algoritmo. Otra razón para considerar a individuos menos adaptados es que ciertas características de estos pueden ser beneficiosas para las nuevas generaciones [56]. En el presente trabajo, se determinó un mecanismo de selección por ruleta. Algunos mecanismos de selección se presentan a continuación:

- **Ranking lineal:** Se trata de ordenar del peor al mejor los individuos de una población de acuerdo con su valor de aptitud. En este caso, los mejores ranqueados obtienen una mejor probabilidad de ser seleccionados. Mientras se crean nuevas generaciones, los criterios de selección son más rigurosos [60].
- **Aleatoria:** Se seleccionan de forma aleatoria dos padres para la reproducción. Este método no garantiza obtener un resultado óptimo [60].
- **Elitismo:** Los individuos con mejor valoración según la función aptitud pasan a la siguiente generación sin modificarse, para conservar los cromosomas con mejores características. El resto de los individuos pasan por operadores de cruce y mutación [60].
- **Torneo:** Se crean subgrupos de dos o más individuos para comparar el valor obtenido de la función objetivo. Aquel de cada subgrupo con mejor aptitud es el seleccionado para el cruce [60].
- **Ruleta:** Simula el comportamiento de una rueda de ruleta en donde cada cromosoma ocupa una sección de esta y cuya área es correspondiente con el valor de la función aptitud de cada individuo. Para este mecanismo, inicialmente se obtiene la suma total acumulada de los valores de la función fitness de cada cromosoma (S_T). Posteriormente, se genera un número aleatorio (n) entre 0 y S_T. Finalmente, se recorre los cromosomas de la población, sumando los valores de aptitud hasta que el resultado sea mayor o igual a n. En este caso, se selecciona al individuo cuya aptitud permitió alcanzar o superar el valor aleatorio [58].

2.5.2.4.2 Cruce

Este operador parte de dos individuos escogidos tras el proceso de selección, mismos que se denominarán padres y que, mediante la combinación de sus características, permitirán la creación de nuevos cromosomas, denominados hijos [58]. En el presente caso, se consideró un método de cruce aritmético (media) para obtener las nuevas generaciones, debido a que se considera una codificación real. Algunos mecanismos de cruce son:

- **Cruce Plano:** Se generan dos descendientes basados en (2.30). En este caso, x_i^1 y x_i^2 son cada uno de los genes de los padres 1 y 2, en el locus i; y λ_i^k es un número aleatorio cambiante entre 0 y 1. Donde $k = 1{,}2$; define a los dos descendientes resultantes del cruce (H) [56].

$$h_i^k = \lambda_i^k x_i^1 + \left(1 - \lambda_i^k\right) x_i^2 \quad ; \quad H_k = \{h_1^k, h_2^k, h_3^k, \dots\} \tag{2.30}$$

- **Cruce Lineal:** Se generan tres hijos, basados en (2.31). Los dos descendientes con mayor adaptación formarán la nueva generación. En este caso $k = 1{,}2{,}3$ [56].

$$h_i^1 = \frac{1}{2} x_i^1 + \frac{1}{2} x_i^2 \quad ; \quad h_i^2 = \frac{3}{2} x_i^1 - \frac{1}{2} x_i^2 \quad ; \quad h_i^3 = -\frac{1}{2} x_i^1 + \frac{3}{2} x_i^2 \tag{2.31}$$

- **Cruce Simple:** Se genera un número d aleatorio entre 1 y $n - 1$, donde n es el último locus de un cromosoma. En base a esto, los nuevos individuos se generan al intercambiar los genes posteriores al locus obtenido aleatoriamente (d) [56].
- **Cruce de dos puntos:** Similar al cruce simple pero se generan dos números aleatorios para realizar el intercambio, en dos locus distintos del cromosoma [56].
- **Cruce uniforme:** Para cada gen del cromosoma, se genera un número aleatorio con valores 0 o 1. Si el valor corresponde a 0, el descendiente conserva el gen del padre 1. Por otro lado, si el número es 1, el descendiente conserva el gen del padre 2. El segundo descendiente recibe los genes no seleccionados para el primer hijo, conservando las mismas posiciones de cada gen [56].
- **Cruce mediante media geométrica:** Cada gen de la descendencia es el resultado de multiplicar los genes de los padres y obtener su raíz cuadrada [58].
- **Cruce aritmético:** Se realiza una combinación lineal de los padres mediante (2.32). En este caso, λ es un número aleatorio entre 0 y 1. Para obtener un método de cruce equivalente a la media aritmética, se emplea $\lambda = 0.5$. Además, $k = 1{,}2$ [56].

$$h_i^1 = \lambda x_i^1 + (1 - \lambda) x_i^2 \quad ; \quad h_i^2 = \lambda x_i^2 + (1 - \lambda) x_i^1 \tag{2.32}$$

2.5.2.4.3 Mutación

El operador de mutación permite realizar ciertas variaciones en los genes de un cromosoma, de tal manera que se puedan introducir nuevas características a la población. Dichas variaciones se realizan sobre los individuos descendientes y son empleadas para mantener la diversidad de los genes, de tal manera que no se obtenga una convergencia temprana o mínimos locales [60]. En el caso de codificación con números reales, la mutación se basa en seleccionar genes de un cromosoma de forma aleatoria y asignar un nuevo valor aleatorio, tomando en cuenta los límites que puede tomar dicho valor. Típicamente, la probabilidad de mutación de un individuo está entre el 1-3% [46]. En el

presente trabajo se consideró un mecanismo de mutación adaptativo, en donde las características de mutación varían de acuerdo con la aptitud de los individuos que conforman cada generación.

2.6 OPTIMIZACIÓN DEL CONTROLADOR BASADO EN LYAPUNOV MEDIANTE ALGORITMOS GENÉTICOS

Para poder optimizar los parámetros del controlador para el seguimiento de trayectorias basado en Lyapunov, se emplea el toolbox de Matlab “Optimize Live Editor”, mismo que permite aplicar un algoritmo genético, cuyo objetivo principal es minimizar la función objetivo mostrada en (2.29). Para cumplir con lo mencionado, se realiza la configuración de parámetros del AG, tal como se mostrará en secciones posteriores.

2.6.1 OPTIMIZE LIVE EDITOR

“Optimize Live Editor” es una herramienta que cuenta con una interfaz visual que permite configurar y aplicar distintos métodos de resolución de problemas de optimización. Esta permite trabajar con funciones objetivo lineales o no lineales, sujetas a límites; y que permiten resolver sistemas no lineales. Una de las principales ventajas que presenta esta herramienta es que las configuraciones realizadas en la interfaz son traducidas de forma automática a código de Matlab para poder ejecutar el programa en un live script [61]. En la Figura 2.8 se puede evidenciar el interfaz de “Optimize Live Editor” empleada para el presente trabajo.

2.6.2 PARÁMETROS DEL AG

Para la implementación del algoritmo genético, y como se observa en la Figura 2.8, se considera la minimización de la función no lineal mostrada en (2.29), tomando en cuenta límites superiores e inferiores para los parámetros a optimizar. Con respecto a estos, se tiene tres parámetros del controlador mostrado en (2.24) y (2.25) que se deben optimizar: K_1, K_2 y q_2. El conjunto de estos tres valores conforma un individuo para el AG en forma de un vector de tres dimensiones. Además, debe mencionarse que los límites de dichos parámetros fueron obtenidos de manera heurística, evaluando los valores máximos y mínimos a partir de los cuales el sistema de un robot móvil cumple con el seguimiento de las trayectorias propuestas.

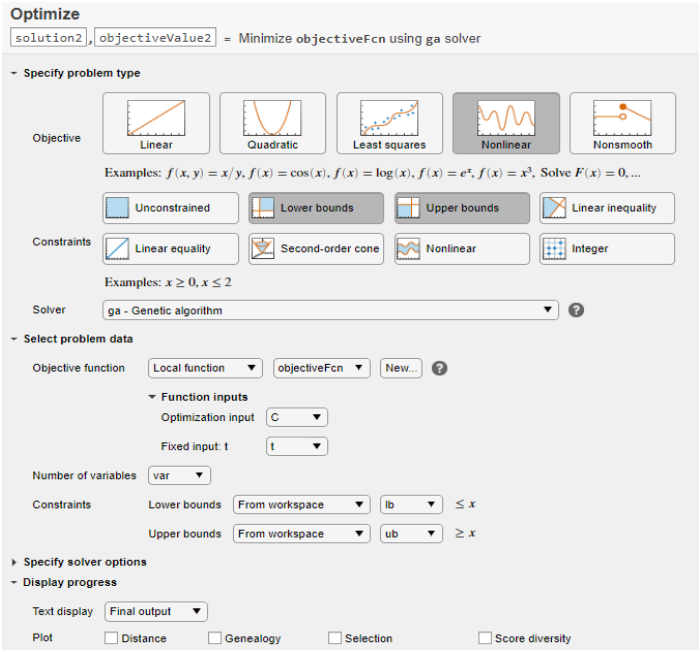

Figura 2.8. Interfaz de “Optimize Live Editor”

Adicionalmente a lo mencionado, se configuraron opciones disponibles en la interfaz, fundamentales para la operación del algoritmo genético, mismas que se muestran a continuación (una explicación más detallada de cada parámetro del AG se presenta en ANEXOS [62]):

Tabla 2.2. Configuración de parámetros para el AG

Parámetro del AG	**Valor**
Creation Function	gacreationuniform (generación aleatoria con distribución uniforme)
Crossover Function	crossoverarithmetic (Cruce aritmético)
Crossover Fraction	0.85
Mutation Function	mutationadaptfeasible (mutación adaptativa factible)
Selection Function	selectionroulette (Selección por ruleta)
Number of elite members	0.05*PopulationSize
Initial population range	[0;25]
Population size	200
Max generations	150
Stall generation limit	50
Constraint tolerance	0.01
Function Tolerance	1e-06

2.6.3 VALORES OPTIMIZADOS PARA UN ROBOT MÓVIL

Una vez configurado el algoritmo genético, como se mostró previamente, se procedió a ejecutar la simulación del programa, tomando en cuenta un sistema de control para seguimiento de trayectorias de un robot móvil. A partir de lo mencionado, se obtuvieron los siguientes resultados del algoritmo genético, considerando como referencia una trayectoria lemniscata:

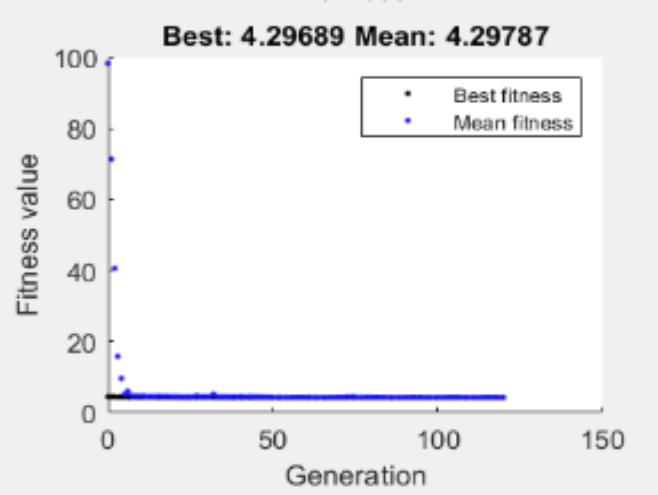

Figura 2.9. Resultado del algoritmo genético (Trayectoria de referencia Lemniscata)

Como se puede evidenciar, el algoritmo genético realiza un proceso iterativo que, para cada ocasión, genera nuevos valores de ganancias para el controlador. Esto a su vez permite obtener nuevos valores para la función objetivo. Una vez cumplida alguna de las condiciones de paro establecidas para el sistema, se adquieren los valores optimizados de los parámetros del controlador. Dichas ganancias obtenidas mediante la optimización con algoritmos genéticos se presentan a continuación:

Tabla 2.3. Ganancias optimizadas para el controlador basado en Lyapunov

Parámetro	Valor
K_1	1.009718203724745
K_2	19.084450710947820
q_2	0.170072491018354

Algunos resultados obtenidos a partir de las ganancias mostradas en la Tabla 2.3 se presentan en la sección 3.1. Sin embargo, dado a que este trabajo no se centra en un robot móvil, dichos resultados solo se mostrarán para una trayectoria de referencia.

2.6.4 ADAPTACIÓN DE VALORES OPTIMIZADOS, PARA SU USO EN UN ROBOT NAO

Los valores mostrados en la Tabla 2.3 permiten un funcionamiento óptimo del controlador para el seguimiento de trayectorias basado en Lyapunov, considerando como planta a un robot móvil diferencial. Sin embargo, si se emplea un robot humanoide NAO, dichos valores no son los adecuados a configurar en el controlador, ya que este robot presenta otras características de movimiento y velocidades máximas en comparación con el robot móvil. Por tal motivo, para poder utilizar los parámetros optimizados mediante AG, es necesario adaptarlos para que se ajusten a las condiciones de operación del robot NAO. Para realizar dicha adaptación, se consideran los límites máximos y mínimos de los parámetros, para un robot móvil; así como los valores máximos y mínimos de dichas ganancias, considerando el uso del humanoide NAO. Estos valores pueden observarse en la Tabla 2.4. En el caso del robot NAO, dichos valores se obtuvieron de forma heurística, sintonizando manualmente el controlador y verificando que este cumpla con el seguimiento de trayectorias planteadas.

Tabla 2.4. Límites de ganancias para el controlador, robot móvil y robot humanoide

Parámetro	Valor mínimo		Valor máximo	
	Móvil	NAO	Móvil	NAO
K_1	1	0.12	6	0.15
K_2	1	0.1	21	2.2
q_2	0.17	0.017	0.25	0.025

A partir de lo mostrado, se obtuvo el porcentaje al que corresponde cada uno de los valores presentados en la Tabla 2.3, respecto al rango de ganancias presentado en la Tabla 2.4. En este caso, los valores mínimos corresponden a un 0% y los valores máximos al 100%. Además, para estos cálculos se consideran los límites de ganancias para el robot móvil.

$$\%_{K_1} = \frac{K_{1_{AG}} - K_{1_{min}}}{K_{1_{max}} - K_{1_{min}}} \cdot 100 = \frac{1.0097 - 1}{6 - 1} \cdot 100 = 0.1944\ \% \tag{2.33}$$

$$\%_{K_2} = \frac{K_{2_{AG}} - K_{2_{min}}}{K_{2_{max}} - K_{2_{min}}} \cdot 100 = \frac{19.0845 - 1}{21 - 1} \cdot 100 = 90.4223\ \% \tag{2.34}$$

En el caso de q_2, se puede evidenciar que sus límites presentan una relación lineal, por lo que no es necesario obtener el porcentaje del valor dentro del rango de máximos y mínimos. En base a los porcentajes calculados, se determinan los valores de las ganancias, para poder aplicarlas al robot NAO:

$$K_{1_{NAO}} = \frac{\%_{K_1} \cdot (K_{1_{max_{NAO}}} - K_{1_{min_{NAO}}})}{100} + K_{1_{min_{NAO}}} \tag{2.35}$$

$$K_{1_{NAO}} = \frac{0.1944 \cdot (0.15 - 0.12)}{100} + 0.12 = 0.1201 \tag{2.36}$$

$$K_{2_{NAO}} = \frac{\%_{K_2} \cdot (K_{2_{max_{NAO}}} - K_{2_{min_{NAO}}})}{100} + K_{2_{min_{NAO}}} \tag{2.37}$$

$$K_{2_{NAO}} = \frac{90.4223 \cdot (2.2 - 0.1)}{100} + 0.1 = 1.9989 \tag{2.38}$$

A partir de los cálculos realizados, se tienen las siguientes ganancias adaptadas para el funcionamiento del robot NAO:

Tabla 2.5. Ganancias adaptadas a partir de valores optimizados

Parámetro	Valor
K_1	0.1200583092
K_2	1.998867325
q_2	0.0170072491

2.7 INTERFAZ GRÁFICA

La interfaz gráfica desarrollada cuenta con cuatro pestañas, tal como se muestra en la Figura 2.10 y Figura 2.11. La primera, denominada INICIO, permite visualizar el título del trabajo de titulación y otros datos respecto al mismo. En la segunda, llamada Instrucciones, se resumen los pasos a seguir para iniciar la simulación del sistema. En la tercera, se puede seleccionar las trayectorias y el controlador a utilizar; además, se muestra una gráfica del movimiento del robot tras completar la simulación. Finalmente, en la cuarta se muestran las gráficas de otras señales, tales como errores y señales de control.

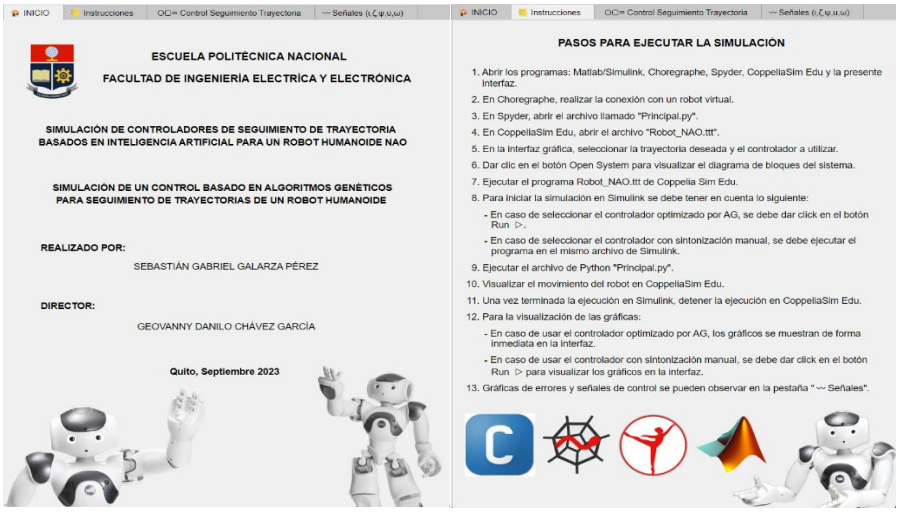

Figura 2.10. Interfaz gráfica (Pestañas Inicio e Instrucciones)

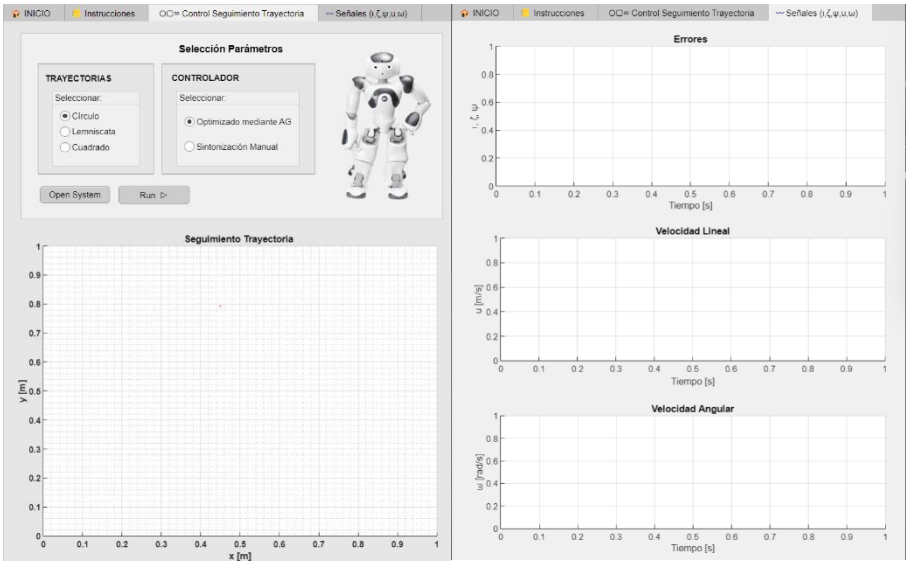

Figura 2.11. Interfaz gráfica (Pestaña Control Seguimiento Trayectorias y Señales)

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

En la presente sección se mostrarán los resultados obtenidos a partir de la optimización de parámetros del controlador. Estos resultados serán mostrados en forma de gráficos y valores. En el caso de los gráficos, se presentan las señales de control, errores, así como las trayectorias de referencia y su comparación con el movimiento real del robot. Por otra parte, se mostrarán los índices de desempeño del sistema para verificar si la optimización de parámetros del controlador permitió reducir dichos valores, lo cual es lo esperado debido a la minimización de la función objetivo mostrada en (2.29).

3.1.1 SEGUIMIENTO DE TRAYECTORIAS PARA UN ROBOT MÓVIL DIFERENCIAL

Los resultados que se mostrarán a continuación son obtenidos considerando un robot móvil diferencial como planta del sistema. En este caso, se consideró como punto de partida del robot a $(x, y) = (3{,}2)\ [m]$, con una orientación $\varphi = \pi/4$. Además, las ganancias del controlador, optimizadas mediante AG, son las mostradas en la Tabla *2.3*.

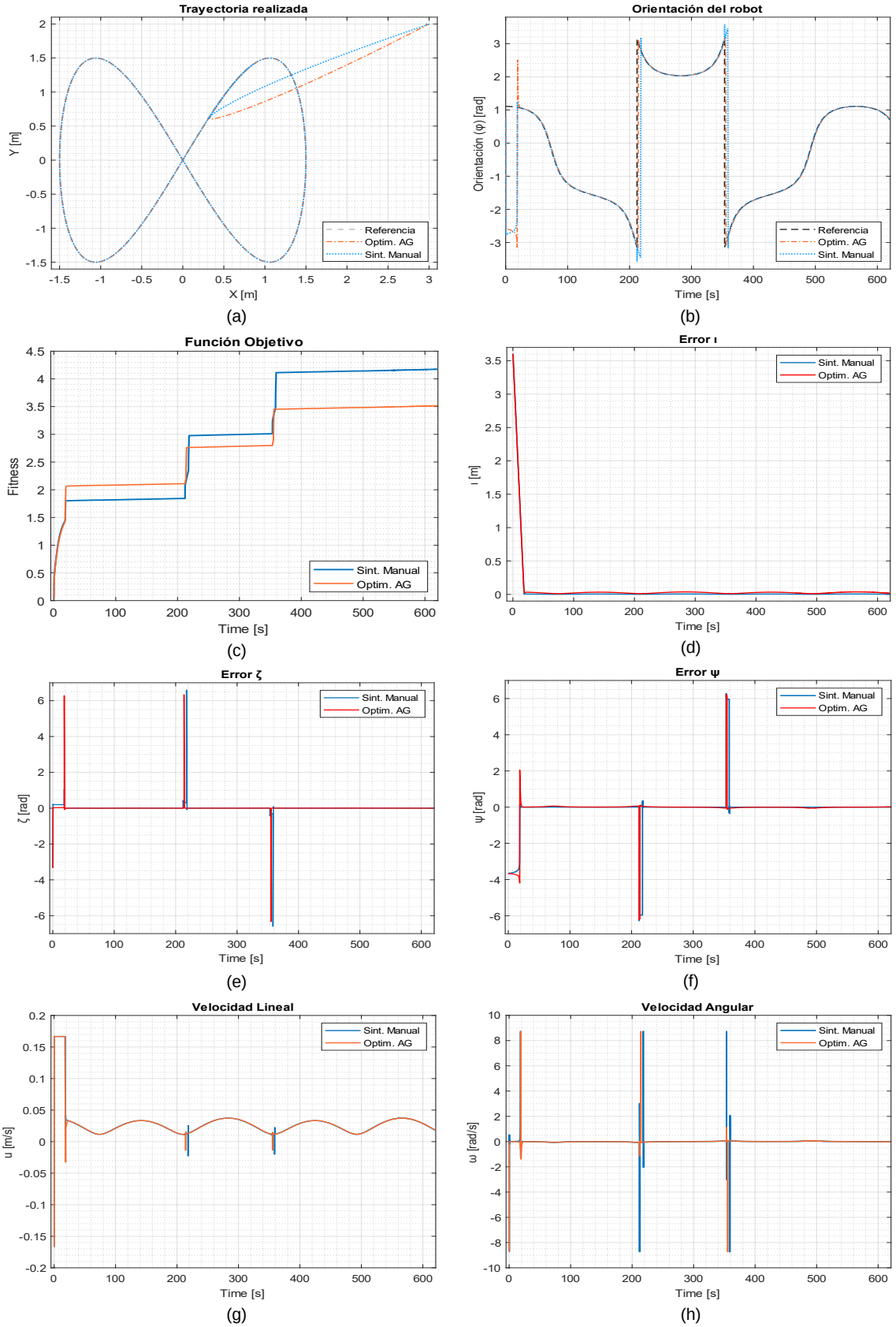

Figura 3.1. (a) Movimiento del robot, (b) Orientación del robot, (c) Función objetivo, (d) Error ι, (e) Error ζ, (f) Error ψ, (g) Velocidad lineal u, (h) Velocidad angular ω

A continuación, se presentan los índices de desempeño del sistema:

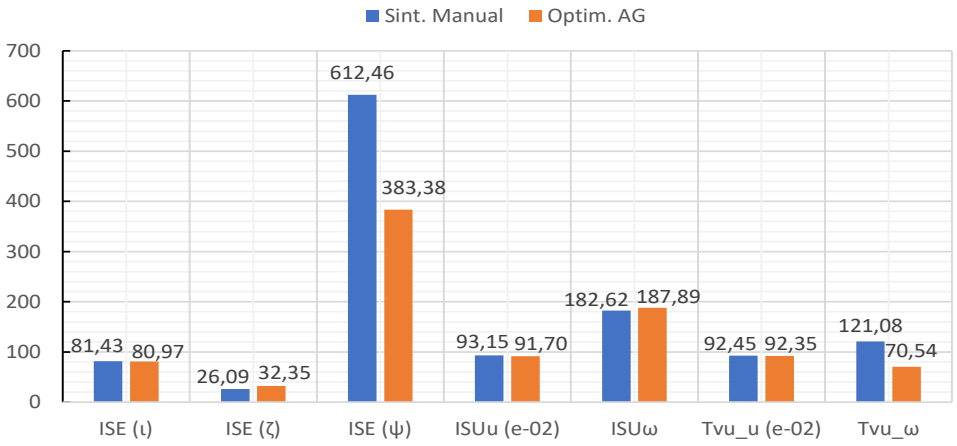

Figura 3.2. Índices de desempeño para trayectoria Lemniscata, Robot Móvil

Como se evidencia, el robot móvil sigue la trayectoria planteada en ambos casos, sin embargo, con la optimización de parámetros por AG se observa un mejor seguimiento de la orientación, señales de control más suaves y una reducción en ciertos tramos de los errores ζ y ψ. Además, se evidencia una reducción del valor de la función objetivo en comparación con lo obtenido para una sintonización manual. Esto también se observa en la Figura 3.2 donde, a pesar de no presentar una reducción en todos los índices, existe una disminución de valores en la mayor parte de ellos, siendo las más considerables en los índices ISE (ψ) y TVu (ω) que presentan una reducción superior al 35% de su valor original. Cabe destacar que el algoritmo genético configurado busca minimizar la función objetivo, misma que es el resultado de una operación entre todos los índices del sistema, más no tiene como finalidad reducir cada uno de los índices de desempeño por separado. Por tal motivo, a pesar de la reducción de valores en la mayor parte de índices, pueden existir algunos de estos que presenten incrementos con respecto al valor calculado mediante el uso de un controlador sintonizado manualmente.

3.1.2 SEGUIMIENTO DE TRAYECTORIAS PARA UN HUMANOIDE NAO

Para el robot NAO, se toma en cuenta un punto de partida $(x, y) = (-0.05, -0.025)\ [m]$, con una orientación $\varphi = 0$ para cada trayectoria de referencia. Para los resultados mencionados, se realiza una comparación de aquellos obtenidos con un controlador sintonizado manualmente, con los adquiridos a partir del controlador cuyas ganancias son las mostradas en la Tabla 2.5.

En la Figura 3.4 y Figura 3.4 se presenta el seguimiento de trayectoria del robot NAO, para una referencia circular de 1.5 m de radio. Como se observa, el controlador con valores optimizados presenta un mejor seguimiento de la orientación, además de una reducción en

la amplitud de los errores ζ y ψ. En cuanto a las señales de control, se evidencia una disminución en la amplitud de la señal de la velocidad angular, lo cual es beneficioso para los actuadores por tratarse de una forma de onda oscilante. Además, se puede observar una reducción en el valor de la función objetivo, lo cual verifica que los valores de ganancias adaptados a partir de los valores optimizados permiten reducir los índices de desempeño del sistema. Esto también se observa en la Figura 3.5, donde existe una reducción de los índices mostrados, a excepción de ISE (ι) e ISUu. En este caso, se tiene una mayor reducción de valores en los índices ISUω y TVu (ω), con un porcentaje superior al 45%.

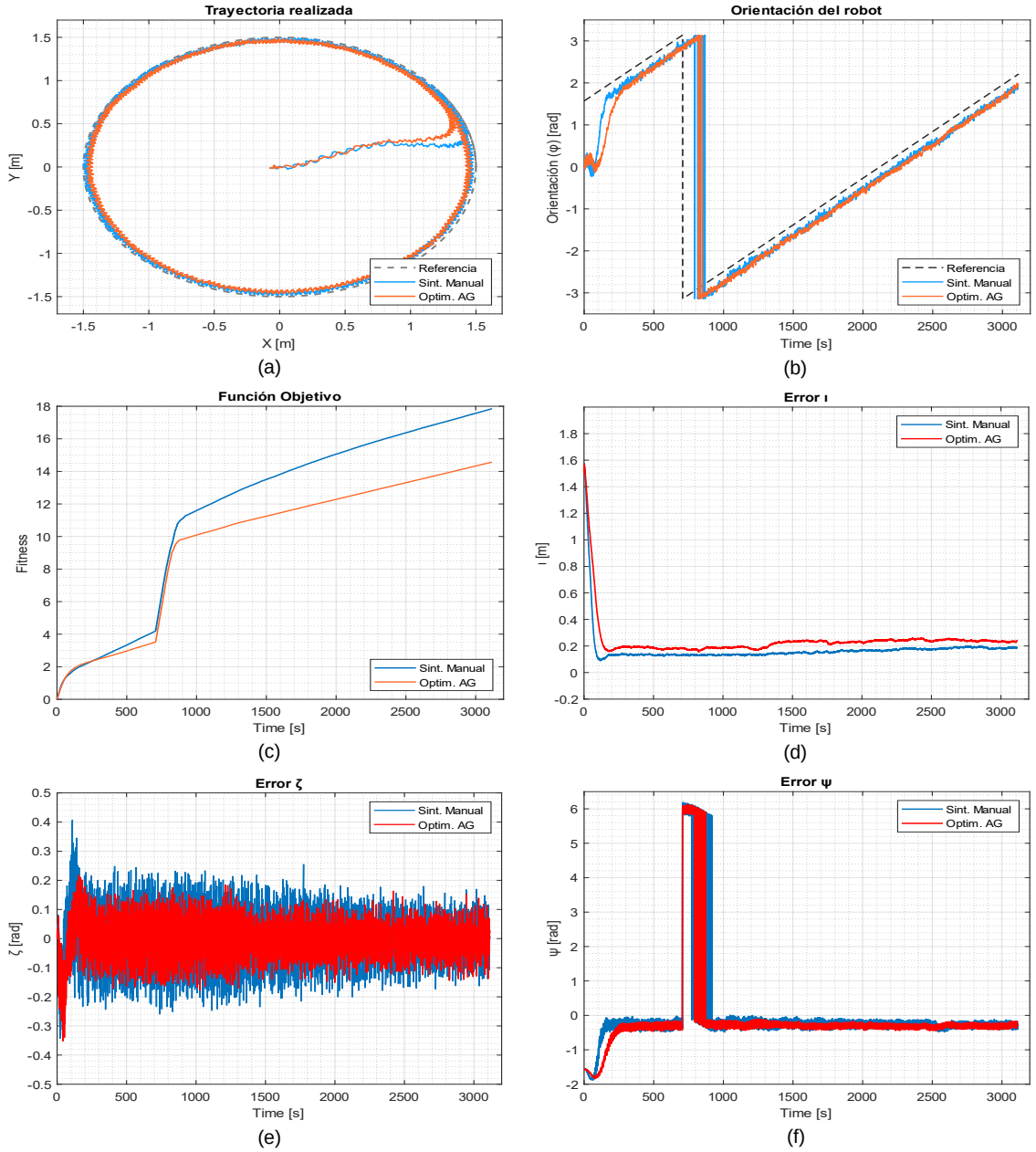

Figura 3.3. Ref. Circular (a) Movimiento del robot, (b) Orientación del robot, (c) Función objetivo, (d) Error ι, (e) Error ζ, (f) Error ψ

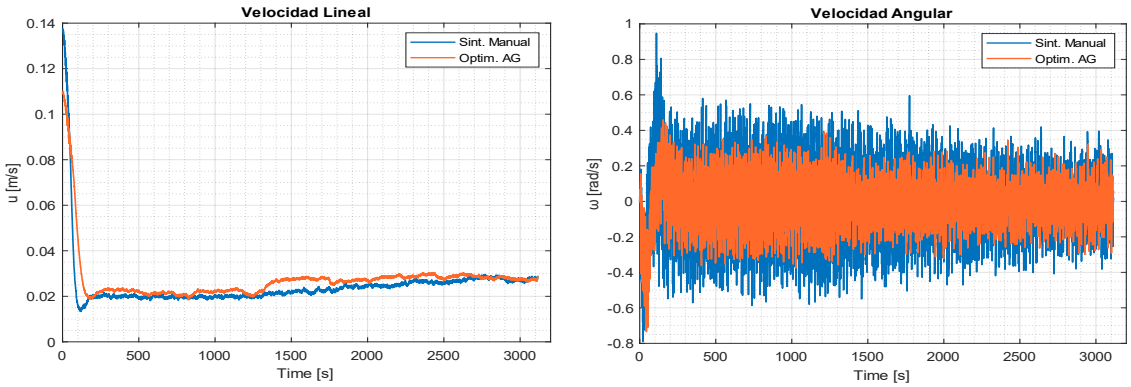

Figura 3.4. Ref. Circular (a) Velocidad lineal u, (b) Velocidad angular ω

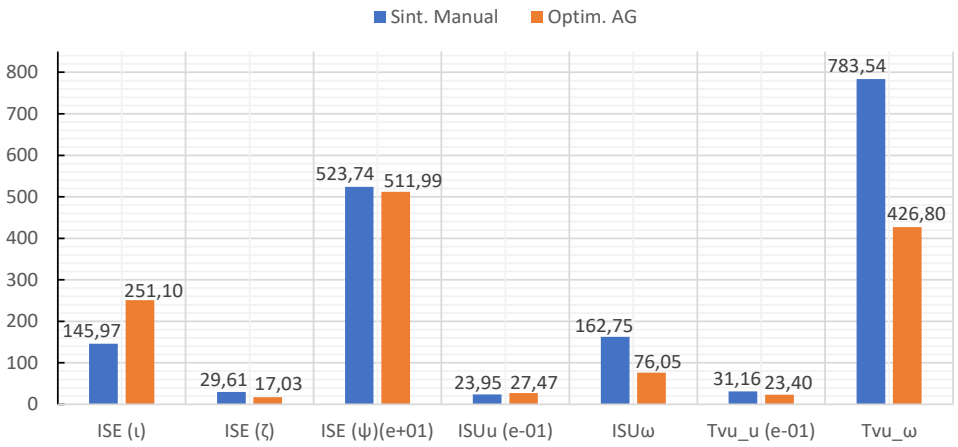

Figura 3.5. Índices de desempeño para trayectoria Circular, Robot NAO

En la Figura 3.6 se muestra el seguimiento de trayectoria del robot NAO, para una referencia en forma de Lemniscata, cuya distancia entre el centro y sus extremos es de 1.5m. Como puede evidenciarse, el controlador con valores optimizados presenta una reducción en la amplitud de los errores ζ y ψ. Además, en cuanto a las señales de control, se observa que la velocidad lineal presenta valores más pequeños en ciertos tramos de la simulación y existe una reducción en la amplitud de la velocidad angular del robot. Ambas características representan un beneficio para los actuadores del robot ya que las velocidades de operación y oscilaciones en las señales se reducen. Por otro lado, la reducción en la función objetivo del sistema muestra que al emplear ganancias optimizadas del controlador, se permite reducir los índices del sistema. Estos índices de desempeño se observan en la Figura 3.7. En este caso, se presenta una disminución en los valores de los índices ISE (ζ), ISUu, ISUω, TVu (u) y TVu (ω); siendo ISE (ζ) e ISUω los que presentan un mayor porcentaje de reducción, con el 28.23% y 41.59% respectivamente.

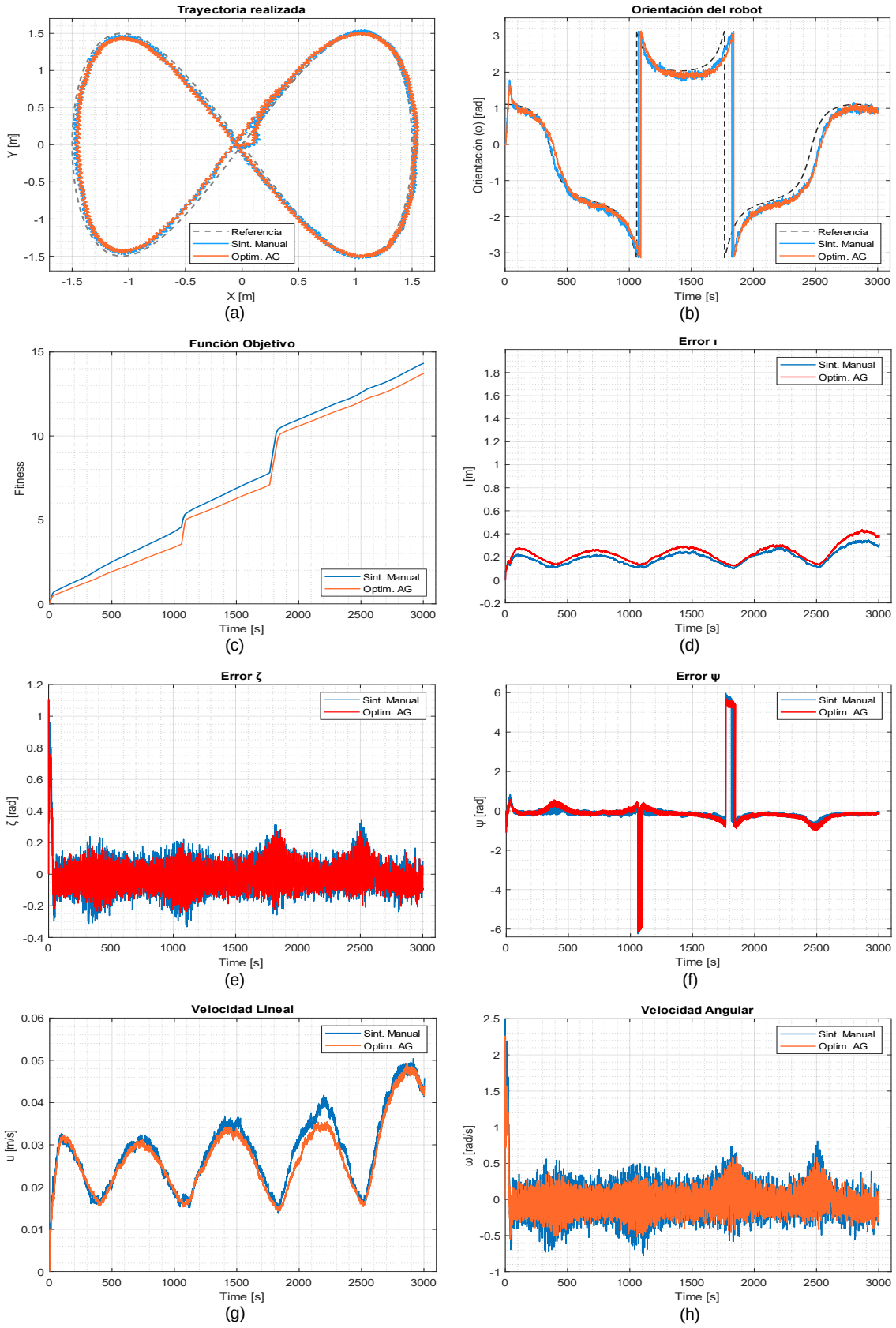

Figura 3.6. Ref. Lemniscata (a) Movimiento del robot, (b) Orientación del robot, (c) Función objetivo, (d) Error ι, (e) Error ζ, (f) Error ψ, (g) Velocidad lineal u, (h) Velocidad angular ω

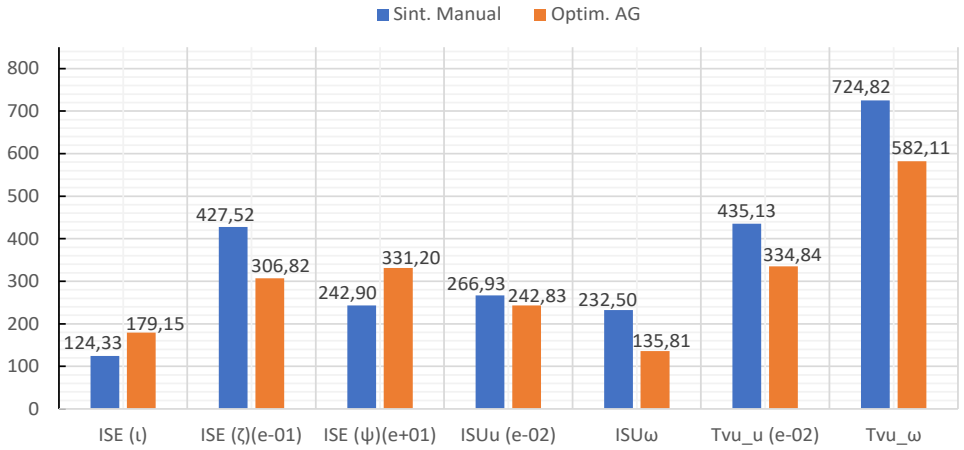

Figura 3.7. Índices de desempeño para trayectoria Lemniscata, Robot NAO

En la Figura 3.8 se muestra el seguimiento de trayectoria para el robot NAO, considerando una referencia cuadrada de 1m de lado. Como puede evidenciarse, en este caso se obtiene una gran reducción en el valor de la función objetivo del sistema, siendo esta disminución mayor a lo mostrado para las otras trayectorias de referencia. Esto se debe a que, con el controlador sintonizado manualmente, se presentan tramos del cuadrado en donde el robot supera los límites de la trayectoria propuesta, lo cual a su vez implica un aumento en los errores del sistema y en las señales de control; esto con el objetivo de que el robot pueda volver a la referencia. Esta característica se puede observar mejor en la Figura 3.8 (g) y Figura 3.8 (h), en donde el controlador sintonizado manualmente presenta velocidades lineales más altas que aquellas mostradas para el sistema con el controlador optimizado; además, en el caso de la velocidad angular, se evidencia que con el controlador sintonizado manualmente se obtienen picos de las señales más elevados en ciertos tramos de la simulación.

Con respecto a los errores, con el control optimizado mediante AG se presenta una reducción en los valores picos de las señales de error ζ y ψ. Además, para el caso mencionado, se puede observar en la Figura 3.8 (b) un mejor seguimiento de la orientación, teniendo un comportamiento más suave en comparación con lo presentado con el controlador con sintonización manual.

En la Figura 3.9 se presentan los índices de desempeño del sistema, a partir de los cuales se puede verificar la razón de la disminución considerable de la función objetivo. Esto se debe a que existe una reducción significativa en prácticamente todos los índices, siendo ISE (ζ), ISE (ψ), ISUu, ISUω y TVu (u) los que presentan un mayor porcentaje de disminución. Dichos porcentajes representan valores del 57.04%, 79.87%, 32.59%, 64.84% y 24.16%, respectivamente.

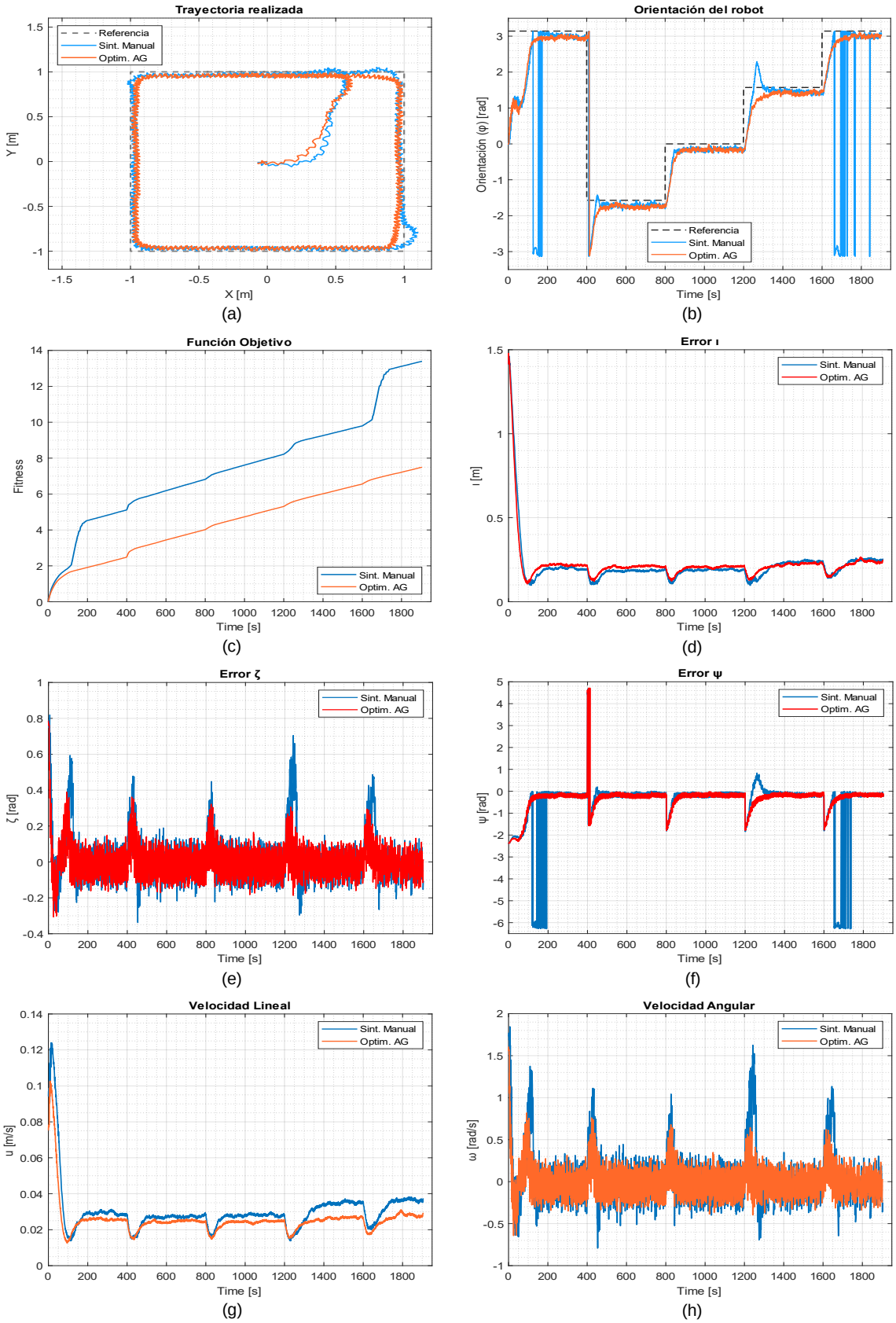

Figura 3.8. Ref. Cuadrada (a) Movimiento del robot, (b) Orientación del robot, (c) Función objetivo, (d) Error ι, (e) Error ζ, (f) Error ψ, (g) Velocidad lineal u, (h) Velocidad angular ω

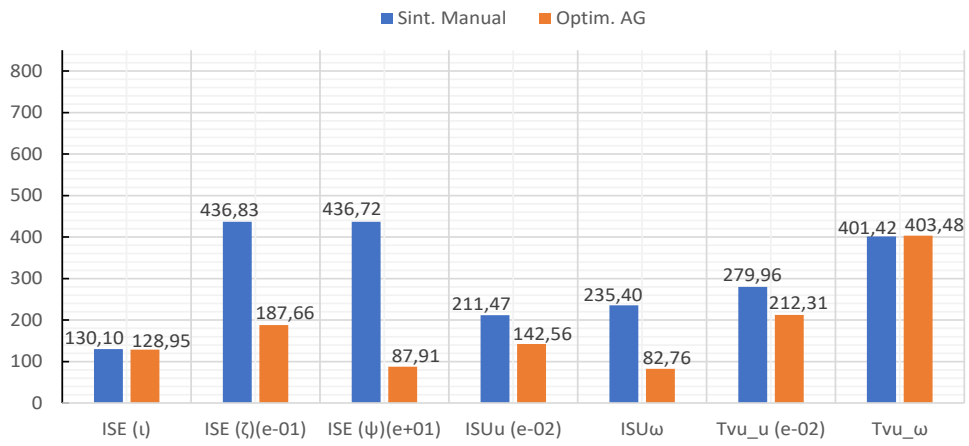

Figura 3.9. Índices de desempeño para trayectoria Cuadrada, Robot NAO

3.2 CONCLUSIONES

- En base a la revisión bibliográfica presentada en este documento, se desarrolló un controlador para el seguimiento de trayectorias basado en Lyapunov, enfocado en un robot móvil, y optimizado mediante algoritmos genéticos; a partir del cual se adaptaron las ganancias del controlador para que pueda realizarse el control de seguimiento de trayectorias en un robot humanoide NAO, de tal manera que este cumpla con el seguimiento de tres trayectorias definidas. Cabe destacar que para el desarrollo del controlador se consideró el uso del modelo cinemático de un robot móvil en configuración diferencial como el modelo matemático que representa el movimiento del humanoide.

- Se realizó la configuración de un API remoto y se estableció comunicación UDP con el objetivo de realizar el envío y recepción de datos entre Matlab y CoppeliaSim Edu, de tal manera que se pueda ejecutar la simulación del sistema. La razón del empleo de comunicación UDP se debe a la necesidad de una transmisión rápida para el envío y recepción de datos, así como la capacidad de enviar datos a múltiples nodos al mismo tiempo. A partir de lo mencionado, se pudo verificar que la comunicación establecida fue la adecuada, permitiendo el funcionamiento esperado del robot en el seguimiento de trayectorias.

- Se pudo observar en los resultados presentados en la sección 3.1.2 que, para cada una de las trayectorias establecidas, se obtuvo una reducción en el valor de la función objetivo del sistema; lo cual es lo deseado para el presente estudio. Esto debido a que el algoritmo genético empleado se configuró de tal manera que este

busque minimizar el valor de dicha función, mediante la optimización de los parámetros del controlador.

- Se puede evidenciar, a partir de los resultados expuestos, que la optimización del controlador mediante algoritmos genéticos permitió mejorar el rendimiento del sistema. Esto se verifica al observar que la gran mayoría de los índices de desempeño calculados redujeron su valor en comparación con lo obtenido mediante una sintonización manual. Esta reducción en los valores de los índices a su vez representa una disminución en los esfuerzos de los actuadores del robot, ya que las señales de control (u y ω) presentaron un comportamiento más suave, con menos oscilaciones o redujeron sus valores al emplear el controlador optimizado. Además, con este controlador se obtuvo, especialmente en las trayectorias cuadrada y circular, un mejor seguimiento de la orientación, tal como se expuso en la sección 3; así como una disminución en la amplitud de los errores ζ y ψ.

- Se puede observar mediante lo expuesto en la sección 3, que el uso del modelo cinemático de un robot móvil en configuración diferencial permitió el cumplimiento de los objetivos establecidos. Además, esta consideración ayudó a que el sistema trabaje con un modelo más simple y que representa un menor coste computacional a la hora de realizar la optimización de parámetros del controlador, mediante algoritmos genéticos.

3.3 RECOMENDACIONES

- Es recomendable realizar una revisión bibliográfica sobre los lenguajes y métodos de programación del robot, para poder determinar la forma más adecuada de operar o controlar el mismo. Además, es recomendable estudiar sobre las librerías y funciones que ofrece NAOqi para la operación del robot.

- En el caso del presente proyecto en donde se emplea la versión 2.7 de Python, es necesario disponer de la versión 2022a de Matlab o anteriores para poder implementar la comunicación UDP entre los programas Matlab y Spyder.

- Se recomienda configurar un tiempo de muestreo de 20 ms para que exista un adecuado envío y recepción de datos para el control del robot. Esto debido a que, como se mostró en la sección 1.4.2.1, el robot realiza una actualización de datos de sus sensores en ciclos de 20 ms. Además de lo mencionado, esta configuración también permite el funcionamiento del módulo ALMotion de Naoqi, mismo que

proporciona métodos que ayudan a realizar los movimientos del robot y que opera a 50 Hz.

- Para futuros trabajos de investigación, se puede considerar la incorporación de obstáculos que el robot NAO deberá evitar durante su movimiento. Esto para de evaluar el desempeño del controlador basado en AG frente a nuevas condiciones de operación. Así mismo, se puede considerar el uso de más de una técnica de IA para la obtención del controlador del sistema, como la optimización de un controlador Fuzzy mediante algoritmos genéticos. También, se puede considerar la implementación física del sistema propuesto en el presente documento.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] J. C. Ponce Gallegos *et al.*, *Inteligencia Artificial*, 1era ed. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014.

[2] J. F. Ávila-Tomás, M. A. Mayer-Pujadas, and V. J. Quesada-Varela, "Artificial intelligence and its applications in medicine II: Current importance and practical applications," *Aten Primaria*, vol. 53, no. 1, pp. 81–88, Jan. 2021, doi: 10.1016/J.APRIM.2020.04.014.

[3] Google Cloud, *¿Qué es la inteligencia artificial o IA?* Accessed: May 18, 2023. [Online]. Available: https://cloud.google.com/learn/what-is-artificial-intelligence?hl=es-419

[4] P. Ponce Cruz, *Inteligencia artificial: con aplicaciones a la ingeniería* , 1era ed. México D.F.: Alfaomega Grupo Editor, 2010.

[5] G. Almeida, "FUNDAMENTOS GENERALES DE LA ROBOTICA," vol. 1, pp. 3–5, 2009.

[6] J. Ruiz de Garibay Pascual, "Robótica: Estado del arte," Universidad de Deusto, Bilbao.

[7] V. R. Barrientos Sotelo, J. R. García Sánchez, and R. Silva Ortigoza, "Robots Móviles: Evolución y Estado del Arte," *Polibits*, no. 35, pp. 12–17, 2007.

[8] D. A. Escobar Gómez, "Diseño e implementación de una celda colaborativa robotizada mediante robots móviles y humanoides para clasificación de objetos," Universidad de las Fuerzas Armadas - ESPE, Sangolquí, 2020.

[9] C. Balaguer Bernaldo de Quirós, "Robots Humanoides," *Revista de Ciencias y Humanidades - Fundación Ramón Areces*, no. 23, pp. 53–58, Jul. 2020, Accessed: May 23, 2023. [Online]. Available: https://www.fundacionareces.es/recursos/doc/portal/2020/07/15/fra-num23-julio-2020.pdf

[10] T. Haidegger, P. Galambos, and I. J. Rudas, "Robotics 4.0 - Are we there yet?," *IEEE 23rd International Conference on Intelligent Engineering Systems (INES)*, pp. 117–124, Apr. 2019, doi: 10.1109/INES46365.2019.9109492.

[11] T. Bajd, M. Mihelj, J. Lenarčič, A. Stanovnik, and M. Munih, *Robotics; Intelligent Systems, Control and Automation: Science and Engineering*, vol. 43. Springer Netherlands, 2010.

[12] M. Ben-Ari and F. Mondada, *Elements of Robotics*. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-62533-1_1/COVER.

[13] A. Dobra, "General classification of robots. Size criteria," *23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, vol. 23, pp. 1–6, Sep. 2014, doi: 10.1109/RAAD.2014.7002249.

[14] S. G. Tzafestas, *Introduction to mobile robot control*, 1st ed. Athens: Elsevier , 2014.

[15] M. A. Zarco López, "Evaluación de técnicas de control visual monocular para la locomoción de robots humanoides NAO ," Universidad Nacional Autónoma de México, México D.F., 2015.

[16] S. M. Orozco-Soto and J. M. Ibarra-Zannatha, "Control con rechazo activo de perturbaciones para el equilibrio de robots humanoides," *Research in Computing Science*, vol. 135, pp. 159–171, May 2017.

[17] F. R. Calvopiña Iglesias and P. E. Valladares Romero, “Interpretación de expresiones faciales en adultos mayores utilizando la visión artificial del robot humanoide NAO,” Universidad Politécnica Salesiana, Quito, 2017.

[18] J. M. Pardos Gotor, “Algoritmos de Geometría Diferencial para la Locomoción y Navegación Bípedas de Robots Humanoides Aplicación al robot RH0,” Universidad Carlos III de Madrid, Leganés, 2005.

[19] D. Gomez Rivera, “Medición y modelado bilineal del Zero Moment Point (ZMP) mediante la técnica de subespacios recursiva,” Benemérita Universidad Autónoma de Puebla, Puebla, 2016.

[20] M. Cognetti, “Motion planning for manipulation and/or navigation tasks with emphasis on humanoid robots,” Sapienza - Universita di Roma, Roma.

[21] L. Bascetta *et al.*, “Towards safe human-robot interaction in robotic cells: an approach based on visual tracking and intention estimation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 1–4.

[22] A. Paolillo, A. Faragasso, G. Oriolo, and M. Vendittelli, “Vision-based maze navigation for humanoid robots,” *Auton Robots*, vol. 40, no. 2, pp. 293–309, Feb. 2016, doi: 10.1007/s10514-015-9533-1.

[23] Aldebaran Robotics, “SoftBank Robotics Documentation,” 2019. http://doc.aldebaran.com/ (accessed Nov. 13, 2022).

[24] Revista de Robots, “Robot NAO para Empresa y Educación,” Aug. 11, 2020. https://revistaderobots.com/robots-y-robotica/robot-nao-caracteristicas-y-precio/ (accessed Nov. 13, 2022).

[25] Aldebaran - United Robotics Group, “NAO 6,” 2022. https://www.aldebaran.com/en/nao (accessed Nov. 14, 2022).

[26] SoftBank Robotics, “Specifications_NAO6.” Accessed: Nov. 14, 2022. [Online]. Available: https://www.generationrobots.com/media/Specifications_NAO6.pdf

[27] Y. Kali, M. Saad, J. F. Boland, J. Fortin, and V. Girardeau, “Walking task space control using time delay estimation based sliding mode of position Controlled NAO biped robot,” *Int J Dyn Control*, vol. 9, no. 2, pp. 679–688, Jun. 2021, doi: 10.1007/s40435-020-00696-x.

[28] S. Shamsuddin *et al.*, “Humanoid robot NAO: Review of control and motion exploration,” *IEEE International Conference on Control System, Computing and Engineering*, pp. 511–516, 2011, doi: 10.1109/ICCSCE.2011.6190579.

[29] L. Rouhiainen, *INTELIGENCIA ARTIFICIAL: 101 COSAS QUE DEBES SABER HOY SOBRE NUESTRO FUTURO*, 1st ed. Barcelona, 2018.

[30] R. Benítez, A. Cencerrado-Barraqué, G. Escudero, and S. Kanaan, *Inteligencia artificial avanzada*, 1st ed. 2013.

[31] F. Escolano-Ruiz, M. Á. Cazorla-Quevedo, M. I. Alfonso-Galipienso, O. Colomina-Pardo, and M. Á. Lozano-Ortega, *Inteligencia artificial: modelos, técnicas y áreas de aplicación*. Magallanes, 2003.

[32] I. J. Nagrath and M. Gopal, *Control Systems Engineering*, 4th ed. Delhi: New Age International Publishers, 2006.

[33] N. S. Nise, *Control Systems Engineering*, 7th ed. 2015.

[34] R. C. Dorf and R. H. Bishop, *Sistemas de Control Moderno*, 10th ed. Madrid: Prentice Hall, 2005.

[35] B. C. Kuo, *Sistemas De Control Automático*, 7th ed. Naucalpan de Juárez: Prentice Hall Hispanoamericana, 1996.

[36] L. E. Solaque-Guzmán, M. A. Molina-Villa, and E. L. Rodríguez-Vásquez, "Seguimiento de trayectorias con un robot móvil de configuración diferencial," *Ing. USBMed*, vol. 5, no. 1, pp. 26–34, Jun. 2014, doi: https://doi.org/10.21500/20275846.298.

[37] A. I. Yandún-Torres, "Planeación y seguimiento de trayectorias para un robot móvil," Escuela Politécnica Nacional, Quito, 2011.

[38] M. A. Ojeda-Misses, "Aplicación con el robot Cozmo para el seguimiento de trayectorias como un sistema embebido," *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, vol. 10, no. 6, pp. 24–32, Nov. 2022, doi: https://doi.org/10.29057/icbi.v10iEspecial6.8873.

[39] L. F. Vázquez-Alberto, "Seguimiento de Trayectorias para un robot bípedo por modos deslizantes continuos," Universidad Nacional Autónoma de México, México D.F., 2016.

[40] E. R. Nuñez Quishpe and J. C. Ortega Quezada, "Diseño, simulación y comparación de un controlador pid y un controlador basado en lyapunov para el desplazamiento de un robot humanoide nao v6 sobre un camino generado por un algoritmo de exploración rápida de árbol aleatorio -rrt," Escuela Politécnica Nacional, Quito, 2021.

[41] G. Oriolo, A. Paolillo, L. Rosa, and M. Vendittelli, "Vision-based trajectory control for humanoid navigation," *IEEE-RAS International Conference on Humanoid Robots*, no. 13, pp. 118–123, 2013, doi: 10.1109/HUMANOIDS.2013.7029965.

[42] O. Regalón-Anias, V. Rodríguez-Diez, M. Diez-Rodríguez, and R. Báez-Prieto, "Classic, adaptable and robust control algorithm application, to variant parameter dynamic system," *Ingeniería Energética*, vol. XXXIII, no. 3, pp. 184–195, Jul. 2012.

[43] O. Camacho, A. Rosales, and F. Rivas, *Control de Procesos*, 1st ed. Quito: EPN Editorial, 2020.

[44] M. Santos, "Un Enfoque Aplicado del Control Inteligente," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 8, no. 4, pp. 283–296, Oct. 2011, doi: 10.1016/J.RIAI.2011.09.016.

[45] R. Galán, A. Jiménez, R. Sanz, and F. Matía, "Control Inteligente," *Revista Iberoamericana de Inteligencia Artificial*, vol. 4, no. 10, pp. 43–48, 2000.

[46] E. I. Salazar Zambrano, "Análisis comparativo de técnicas de optimización para la sintonización de controladores PID adaptativos," Escuela Politécnica Nacional, Quito, 2020.

[47] A. J. Enríquez Salinas, “Diseño y simulación de un sistema de teleoperación de un robot humanoide NAO : Diseño y simulación del control de posicionamiento de un robot humanoide NAO.,” Escuela Politécnica Nacional, Quito, 2022.

[48] V. M. Alfaro and R. Vilanova, “Sintonización de los controladores PID de 2GdL: desempeño, robustez y fragilidad,” in *XIV Congreso Latinoamericano de Control Automático (CLCA)*, Santiago, Aug. 2010, pp. 267–272.

[49] J. Alonso González, “Simulación de Sistemas de control continuos con SIMULINK,” Oviedo. Accessed: Jun. 06, 2023. [Online]. Available: http://www.isa.uniovi.es/~alonsog/Regulacion/PL%2006%20Simulacion%20de%20sistemas%20de%20control%20continuo%20con%20SIMULINK.pdf

[50] Coppelia Robotics, “Robot simulator CoppeliaSim: create, compose, simulate, any robot.” https://www.coppeliarobotics.com/ (accessed Jun. 06, 2023).

[51] Spyder Project Contributors, “Spyder IDE,” 2022. https://www.spyder-ide.org/ (accessed Jun. 07, 2023).

[52] Aldebaran Robotics, “Choregraphe overview — NAO Software 1.14.5 documentation.” http://doc.aldebaran.com/1-14/software/choregraphe/choregraphe_overview.html (accessed Jun. 07, 2023).

[53] D. E. Hernández Sánchez, J. R. Eguibar Cuenca, C. Cortés Sánchez, and J. F. Reyes Cortés, “‘Diseño, construcción y modelo dinámico de un robot móvil de tracción diferencial aplicado al seguimiento de trayectorias,’” in *XXIII Congreso Internacional Anual de la SOMIM*, Cuernavaca, Sep. 2017, pp. 1–7.

[54] K. Ogata, *Ingeniería de Control Moderna*, 3rd ed. Naucalpan de Juárez: Pearson Prentice Hall, 1998.

[55] E. Cruz Zavala, “Funciones de Lyapunov de control para el diseño de controladores discontinuos,” Universidad Nacional Autónoma de México, México D.F., 2014.

[56] J. Á. Fernández Prieto, “Optimización Evolutiva de los Parámetros de Control de un Algoritmo Genético,” Universidad de Alcalá, Alcalá de Henares, 2009.

[57] A. López Martínez, “Construcción de un modelo de memoria visual para la navegación de robots humanoides,” Centro de Investigaciones en Óptica, León Gto, 2014.

[58] M. F. Panchi Herrera, “Optimización de Controladores Digitales PID en sistemas dinámicos usando Algorítmos Genéticos,” Escuela Politécnica Nacional, Quito, 2012.

[59] D. A. Tibaduiza Burgos, “Planeamiento de trayectorias de un Robot Móvil,” Universidad Industrial de Santander, Bucaramanga, 2006.

[60] V. del C. Orosco Orozco, “Desarrollo de una herramienta computacional para la sintonización de parámetros de controladores PID y SMC para el seguimiento de trayectoria de un cuadricóptero basado en Algoritmos Genéticos,” Escuela Politécnica Nacional, Quito, 2018.

[61] The MathWorks Inc., “Optimize in the Live Editor - MATLAB Documentation,” 2023. https://la.mathworks.com/help/matlab/ref/optimize.html (accessed Jun. 25, 2023).

[62] The MathWorks Inc., “ga - Find minimum of function using genetic algorithm - MATLAB Documentation,” 2023. https://la.mathworks.com/help/releases/R2021a/gads/ga.html (accessed Jun. 25, 2023).

ANEXOS

ANEXO I. Definición de Parámetros empleados para el AG

En la Tabla I.1 se muestra una descripción de cada uno de los parámetros configurados para el funcionamiento del algoritmo genético, así como otras opciones disponibles a configurar [62]. Los parámetros empleados para el presente proyecto se pueden observar en la Tabla 2.2.

Tabla I.1. Parámetros para configuración del AG

Parámetro del AG	Descripción
Creation Function	Función que permite la creación de la población inicial. Entre las opciones de esta función se encuentran: • gacreationuniform: Crea una población inicial aleatoria, con una distribución uniforme. • gacreationlinearfeasible: crea una población aleatoria que satisface límites y restricciones lineales. Existe una buena dispersión de individuos en la población generada. • gacreationnonlinearfeasible: es una función predeterminada para el algoritmo “penalty” de restricción no lineal.
Crossover Function	Función que permite crear hijos en una población mediante mecanismos de cruce. Estos pueden ser: • crossoverscattered: función predeterminada para problemas sin restricciones. Crea un vector binario en donde los genes son seleccionados de cada padre, los valores 1 provienen del primer padre y los valores 0 provienen del segundo padre. • crossoversinglepoint: se selecciona un número entero n aleatorio entre 1 y el número de variables del problema. El hijo se forma mezclando los valores ubicados en locus menores a n, del primer padre; y los valores ubicados en locus mayores a n, del segundo padre. • crossovertwopoint: se obtienen dos números aleatorios (n y m). Los valores ubicados en locus

	menores a n se seleccionan del primer padre, los valores ubicados en locus entre n y m se seleccionan del segundo padre; y aquellos superiores a m se seleccionan del primer padre. • crossoverintermediate: función predeterminada para restricciones lineales donde los hijos son un promedio ponderado de los padres. Los pesos se definen mediante el parámetro "ratio". ($hijo = padre1 + rand * ratio * (padre2 - padre1)$). • crossoverheuristic: los hijos se encuentran dentro de la línea que separa los valores de ambos padres, a una distancia pequeña del padre con mejor valor en la función aptitud. Esta distancia entre el mejor padre se especifica con el parámetro "ratio". ($hijo = padre2 + ratio * (padre1 - padre2)$). • crossoverarithmetic: los hijos son la media aritmética ponderada de los padres. La descendencia se obtiene acorde a restricciones y límites lineales.
Crossover Fraction	Define la fracción de la población que es creada mediante la función de cruce, sin incluir a la descendencia de elite. El valor por defecto es 0.8.
Mutation Function	Esta función especifica cómo el AG realiza cambios aleatorios en los individuos para crear hijos mutados. La mutación otorga diversidad a la población. Entre las opciones de mutación están: • mutationgaussian: función predeterminada para problemas sin restricciones. Esta función agrega un número aleatorio tomado de una distribución gaussiana con media 0, a cada gen del individuo padre. • mutationuniform: se selecciona una fracción del individuo a mutar. Cada gen del individuo tiene una tasa "rate" de mutar. Una vez seleccionado, el algoritmo reemplaza esos genes por valores

	aleatorios dentro del rango de ese gen. El valor por defecto de mutación es 0.01. • mutationadaptfeasible: función por defecto cuando existen restricciones. Se genera aleatoriamente una dirección hacia la que se modificaran los valores del individuo. Dichas direcciones se adaptan de acuerdo con la última generación exitosa o fallida. La mutación determina esta dirección y la longitud del paso de cambio de los valores, de acuerdo con límites y restricciones establecidos.
Selection Function	Esta función permite determinar cómo el AG elige a los padres para producir la siguiente generación. Entre las opciones de selección se encuentran: • selectionstochunif: Con esta función se traza una línea en la que cada padre representa una sección de la línea, de una longitud proporcional a su valor escalado. Inicialmente, el algoritmo genera un número aleatorio n para el tamaño del paso. A partir de ello, se recorre la línea en pasos de igual tamaño, asignando un padre de cada sección en la que se detiene. • selectionremainder: los padres son seleccionados de forma determinista, a partir de la parte entera del valor escalado de cada individuo. Posteriormente, usa un mecanismo de selección por ruleta para la parte fraccionaria restante. • selectionuniform: los padres son seleccionados en base a las expectativas y el número de padres. • selectionroulette: Los padres son elegidos mediante la simulación de una rueda de ruleta. En este caso, el área de cada sección de la rueda correspondiente a un individuo es proporcional a la expectativa de este. • selectiontournament: inicialmente, se determina una cantidad “size” de candidatos de forma

	aleatoria. Posteriormente, se selecciona al mejor individuo del grupo para ser padre para la siguiente generación. Size debe tomar un valor mayor a 2, siendo el valor por defecto 4.
Number of elite members	Variable que define cuántos individuos de la población actual se garantiza que prevalezcan para la próxima generación. El valor por defecto es 0.05*PopulationSize
Initial population range	Se trata de una matriz o vector que especifica el rango que tendrán los individuos en la población inicial. Se aplica para la función gacreationuniform.
Population size	Define el tamaño que tendrá la población
Max generations	Define el número máximo de iteraciones antes de que se detenga el algoritmo.
Stall generation limit	Define el número de generaciones que deben transcurrir sin que exista un cambio significativo en la función objetivo (menor a Function Tolerance), antes de que el algoritmo se detenga.
Constraint tolerance	Determina la viabilidad respecto a restricciones no lineales. Además, $\max(sqrt(eps), ConstraintTolerance)$ define la viabilidad respecto a restricciones lineales.
Function Tolerance	Si un el valor de la función objetivo es menor a este parámetro, se considera que el cambio relativo promedio de la función objetivo no es significativo. Es empleado para determinar el paro del algoritmo genético.

ANEXO II. Manual de Usuario para la Simulación del Sistema

Para poder realizar la simulación del sistema, es necesario que todos los programas que se mencionaron en la sección 1.4.7 se encuentren instalados en el computador. A continuación, se presentan las versiones de los programas que fueron empleados para el presente trabajo:

- Anaconda2 5.3.0 (Python 2.7.15), dentro del paquete del programa, se cuenta con Spyder 3.3.1.
- MATLAB R2021a
- CoppeliaSim Edu 4.4.0.
- Choregraphe Suite 2.8.7.4.

Para realizar la simulación del sistema, mediante el uso de los programas mencionados, se empleó un computador con las siguientes características:

- Procesador: Intel(R) Core(TM) i7-10750H CPU @ 2.60 GHz, 6 núcleos
- Memoria RAM: 16 GB.
- Sistema Operativo: Windows 11 Home, version 22H2
- Tarjeta gráfica: NVIDIA GeForce RTX 2060

Una vez que los programas se encuentren instalados, se requiere realizar ciertas configuraciones previas, mismas que se presentan a continuación:

a) **Instalación de Python SDK:** se trata de una biblioteca de software que contiene las funciones necesarias para controlar robots de Aldebaran Robotics, empleando Python. Además, ciertas funciones son empleadas internamente por Choregraphe para su funcionamiento. Para su instalación, se deben seguir los siguientes pasos:

    - Descargar el SDK de Python desde la página del fabricante del robot NAO (se puede encontrar también el SDK de Python en la carpeta Instaladores de los archivos adjuntos al presente proyecto).
    - Descomprimir el archivo y copiar las 7 carpetas contenidas en la carpeta de destino, ubicada en la dirección: C:\\Users\NAME\Anaconda2.
    - Abrir “Editar las variables del entorno del sistema”, misma que se puede encontrar desde el buscador de Windows 11.

- Seleccionar la opción “Variables de entorno…” dentro de la pestaña “Opciones Avanzadas”.
- Agregar una nueva variable de usuario llamada *PYTHONPATH*, direccionada a: C:\\Users\NAME\Anaconda2\lib como se muestra en la Figura II.1.

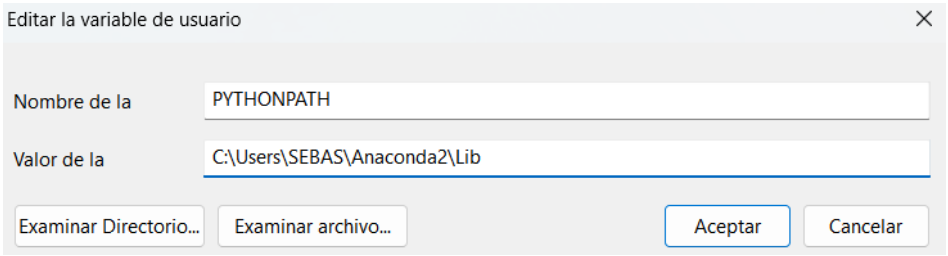

Figura II.1. Creación de una nueva variable de usuario

b) **Configuración de Choregraphe y conexión a Robot Virtual:** Para configurar el modelo del robot NAO, se deben seguir los siguientes pasos:

- Dentro del software Choregraphe, se selecciona la opción “Preferences” de la pestaña “Edit”.
- Seleccionar la pestaña “Virtual Robot” y, en la opción Robot model, se debe buscar el modelo “NAO H25 (V6)”.

Para realizar la conexión con un robot virtual, se deben seguir los siguientes pasos:

- Dar clic en la opción “Connect to…” dentro de la pestaña “Connection”.
- Marcar la opción “Use fixed port (9559)” y posteriormente dar clic en el botón “Select”, como se muestra en la Figura II.2.

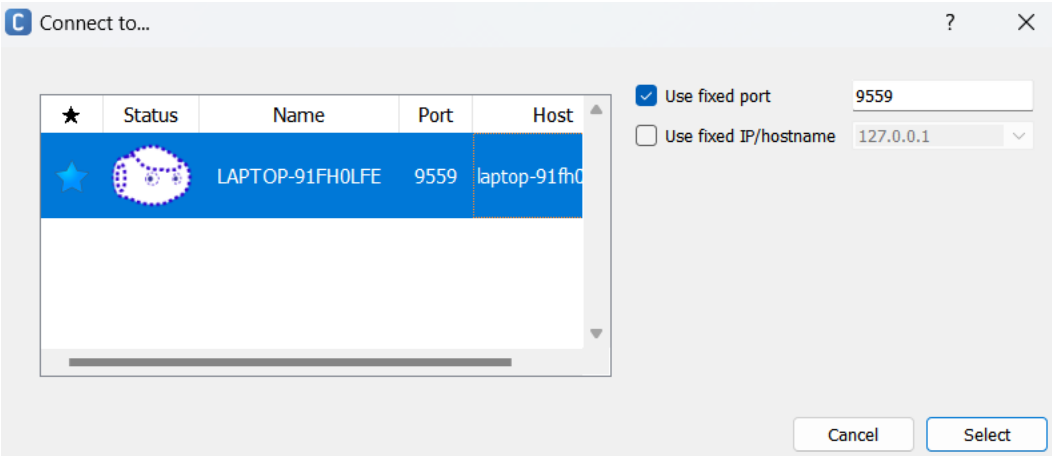

Figura II.2. Conexión a un Robot Virtual

Tras realizar las configuraciones previamente mostradas, se deben seguir los siguientes pasos en el orden mostrado, para poder ejecutar la simulación del sistema:

1. De forma inicial, se deben abrir los programas: Matlab/Simulink, Choregraphe, Spyder y CoppeliaSim Edu.
2. Abrir el archivo correspondiente a la interfaz gráfica desarrollada, mismo que puede encontrarse en los archivos adjuntos al presente proyecto, con el nombre "Interfaz.mlapp".
3. Realizar la conexión con un Robot Virtual en Choregraphe, siguiendo los pasos explicados previamente,
4. En el software Spyder, abrir el archivo denominado "Principal.py". Este permite la ejecución de varias operaciones principales del programa de forma simultánea. En primer lugar, se utiliza la función "initialize()" que permite iniciar la conexión con un servidor API remoto (CoppeliaSim Edu), así como definir las articulaciones del robot en una matriz llamada "Body". Por otro lado, se usa la función "JointControl()" que inicializa cada una de las articulaciones del robot y permite que se produzca movimiento en estas durante la simulación. Finalmente, se emplea la función "Controlador()" en donde se define un posicionamiento inicial de las articulaciones del robot, se obtienen datos de posición, orientación y velocidades procedentes de CoppeliaSim Edu; y se realiza el envío de datos hacia Simulink para cerrar el lazo de control.
5. En el software CoppeliaSim Edu, abrir el archivo denominado "Robot_NAO.ttt". En este se puede encontrar el ambiente de simulación donde se podrá visualizar el movimiento del robot. Dentro de este programa, se ubicó al robot NAO disponible en el software, como se observa en la Figura II.3, así como un cuboide en un espacio no visible. Este último elemento permite inicializar el API remoto que permitirá la comunicación con Python. Para esto, se añadió el código "simRemoteApi.start(19999)" en un script del Cuboide.
6. Dentro de la interfaz gráfica desarrollada, antes de iniciar la simulación, se debe seleccionar la trayectoria deseada, así como el controlador: optimizado mediante AG o sintonizado de forma manual. Un ejemplo de esta selección se puede observar en la Figura II.4.

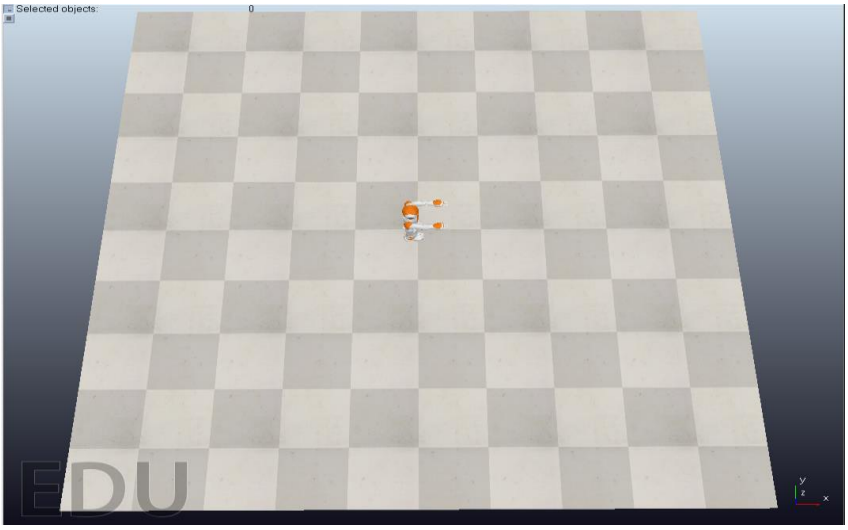

Figura II.3. Entorno de Simulación en CoppeliaSim Edu

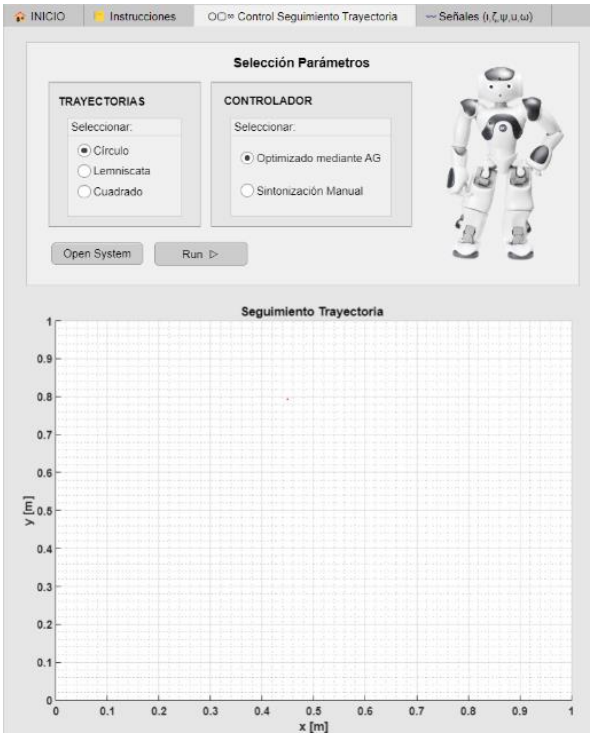

Figura II.4. Selección de Trayectoria y Controlador

7. Una vez realizado el paso previo, se debe dar clic en el botón “Open System” ubicado en la tercera pestaña de la interfaz gráfica, para que se visualice el diagrama de bloques del sistema en Simulink, tal como se observa en la Figura II.5.

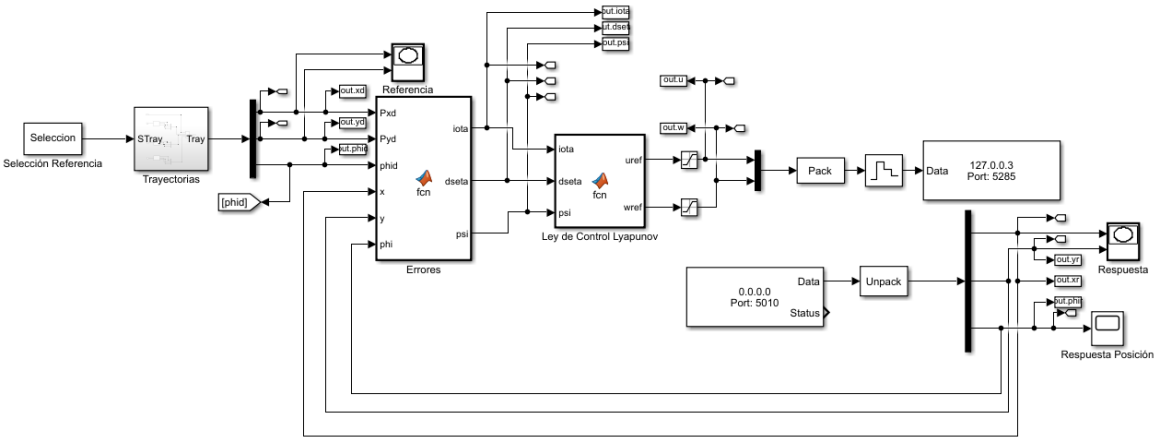

Figura II.5. Diagrama de Bloques del sistema

8. Iniciar la simulación del programa “Robot_NAO.ttt” de CoppeliaSim Edu.
9. Para iniciar la simulación en Simulink se debe realizar las siguientes actividades, de acuerdo con lo seleccionado en el paso 6:
    - En caso de seleccionar el controlador optimizado mediante algoritmos genéticos, dar clic en el botón “Run” ubicado en la tercera pestaña de la interfaz gráfica.
    - En caso de seleccionar el controlador sintonizado manualmente, iniciar la simulación directamente del archivo de Simulink abierto en el paso 7.
10. Ejecutar el archivo de Python “Principal.py” en el software Spyder.
11. Tras realizar los pasos previamente expuestos, comenzará la simulación del sistema. En este caso, el seguimiento de trayectorias del robot NAO puede ser visualizado en el software CoppeliaSim Edu.
12. Una vez que finalice la ejecución de Simulink, se debe detener de forma manual la simulación de CoppeliaSim Edu.
13. Para poder visualizar los resultados de simulación, es necesario considerar lo siguiente:
    - En caso de haber seleccionado el controlador optimizado mediante algoritmos genéticos, los gráficos que exponen los resultados de simulación se mostrarán en la tercera y cuarta pestaña de la interfaz de forma inmediata, tras finalizar la simulación.

- En caso de haber seleccionado el controlador sintonizado manualmente, se debe dar clic en el botón “Run” de la interfaz para visualizar las gráficas mencionadas.

NOTA: En caso de que al finalizar la simulación se observe en Choregraphe al robot NAO “sentado”, se debe iniciar la simulación en el archivo de CoppeliaSim Edu y posteriormente ejecutar el archivo de Python “single_nao_control.py”, disponible en los archivos adjuntos de este proyecto. Esto permite que el robot se levante y esté listo para una siguiente simulación. Si, a pesar de realizar lo mencionado, el robot continúa “sentado”; es necesario cerrar Choregraphe y finalizar las siguientes tareas desde el Administrador de Tareas: “naoqi-service.exe”, “qilaunch.exe” y “qi-secure-gateway.exe”. Tras realizar lo mencionado, se puede abrir nuevamente Choregraphe y seguir los pasos para iniciar la simulación.

NOTA: Previo a realizar los pasos mencionados para la simulación del sistema, es recomendable limpiar el workspace de Simulink mediante el comando “clear”.

En la Figura II.6 y Figura II.7 se puede visualizar, a modo de ejemplo, resultados de simulación dentro de la interfaz gráfica, para una trayectoria de referencia y empleando cada opción de controlador disponible, respectivamente.

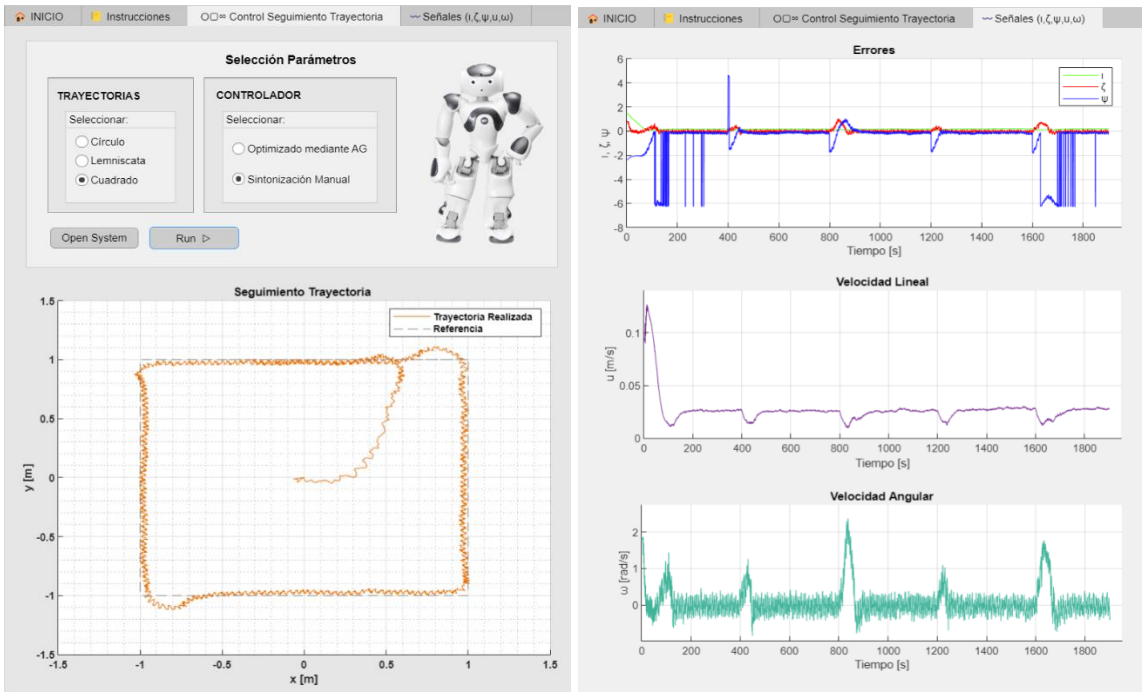

Figura II.6. Resultados de Simulación, Referencia Cuadrado, Controlador sintonizado manualmente

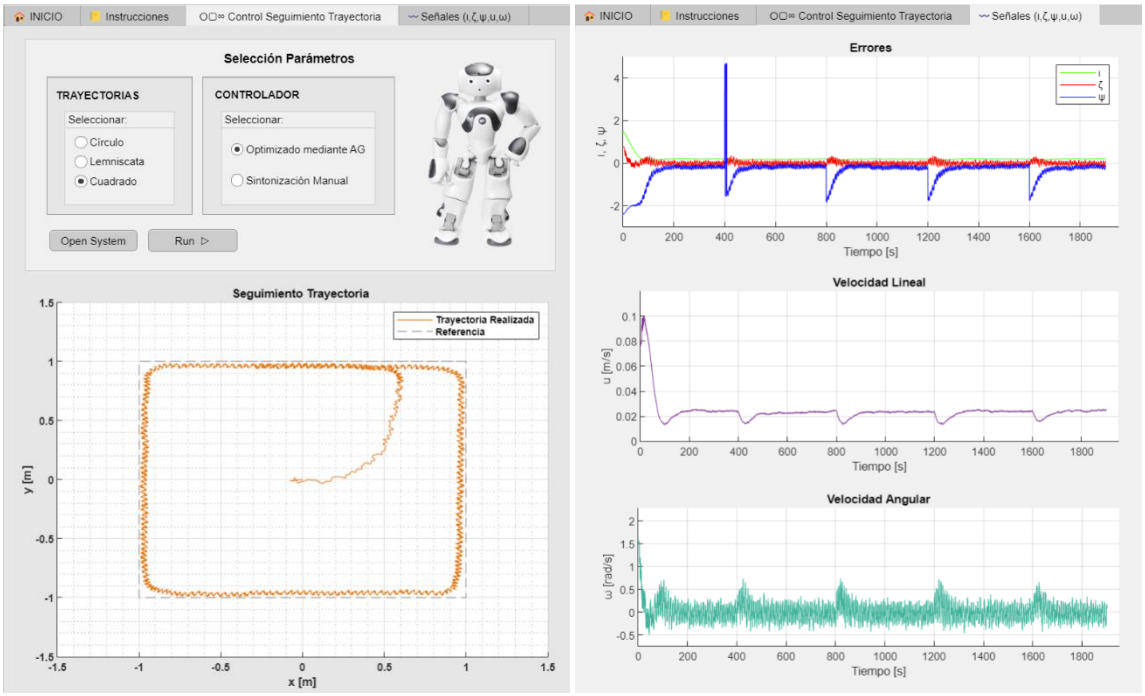

Figura II.7. Resultados de Simulación, Referencia Cuadrado, Controlador optimizado mediante AG