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 LÓGICA DIFUSA PARA 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

JOEL RODRIGO CARRILLO TORRES

joel.carrillo@epn.edu.ec

DIRECTOR: GEOVANNY DANILO CHAVEZ GARCÍA

danilo.chavez@epn.edu.ec

DMQ, septiembre 2023

CERTIFICACIONES

Yo, JOEL RODRIGO CARRILLO TORRES 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.

_____

JOEL RODRIGO CARRILLO TORRES

Certifico que el presente trabajo de integración curricular fue desarrollado por JOEL RODRIGO CARRILLO TORRES, bajo mi supervisión.

_____

GEOVANNY DANILO CHAVEZ GARCÍA
DIRECTOR

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.

JOEL RODRIGO CARRILLO TORRES

GEOVANNY DANILO CHAVEZ GARCÍA

DEDICATORIA

A mí, por no rendirme ante cada oportunidad que se presentó, por ser el autor de este trabajo, y por siempre dar lo mejor de mí. A mi familia, que siempre me motiva a seguir adelante y jamás renunciar. Y a todas las personas que vislumbran un mundo mejor y un futuro prometedor de la mano de la ciencia e innovación tecnológica.

“El problema no es el problema,
el problema es tu actitud frente
al problema.”
-Cap. Jack Sparrow.

AGRADECIMIENTO

Agradezco a Dios, o todo el complejo significado que las personas tenemos de él, por brindarme salud y poner en mi camino a las personas correctas.

A mis padres, Inés y Rodrigo, quienes toda la vida me han dado su amor y apoyo incondicional. A mi hermana, Mayra, mi ejemplo de esfuerzo y perseverancia. Gracias por creer en mí, por sus consejos y enseñanzas, por ser los pilares fundamentales de mi vida y por darme la fuerza y la valentía para seguir mis sueños.

A mis primos, tíos y abuelos, gracias por todos los momentos compartidos y por enseñarme que soy capaz de lograr lo que me proponga.

Gracias a mis amigos Luis, Darwin, Renato y todos los “Tarquicrax” por ser mis hermanos y siempre levantarme cada vez que me sentía por el suelo. A Nicole, Kevin y Sebas C., por siempre estar presentes e impulsarme a seguir adelante y cumplir mis metas. A Sebastián “Yatra”, por siempre emitirme su sabiduría y darme una mano ante las complicaciones de la recta final de este proyecto. A mis amigas y amigos, gracias por las risas, por ser mis soportes y por hacer de la universidad un lugar más bonito.

A Elizabeth, por brindarme su cariño y apoyo en todo momento. Gracias por ser parte de mi vida y por permitirme ser parte de la suya.

A mi tutor, el Dr. Danilo Chávez, gracias por permitirme ser parte de este proyecto, por su paciencia y expresarme su calma con cada conversación.

A la Escuela Politécnica Nacional, por darme la oportunidad de conformar sus aulas y formarme profesionalmente con excelencia y tenacidad. Gracias a aquellos profesores que lograron asertivamente transmitir sus conocimientos y aumentar mi interés por la ciencia, los admiro mucho. A la Dra. Silvana Gamboa, gracias por la calidez de su enseñanza y sus consejos.

A ellos y a todas las personas que, para bien o para mal, se cruzaron en mi camino, gracias por convertirme en la persona que soy actualmente.

ÍNDICE DE CONTENIDO

RESUMEN

En el presente trabajo se expone el diseño y simulación de un controlador basado en inteligencia artificial para el seguimiento de trayectoria de un robot humanoide NAO, utilizando la técnica de lógica difusa. Se propone el desarrollo de un sistema de control a alto nivel del robot NAO, empleando un controlador cinemático tipo PD basado en lógica difusa con compensación de trayectoria, y partiendo del modelo cinemático directo de un robot móvil.

El controlador es implementado en la interfaz de Matlab/Simulink y se establece comunicación mediante UDP con el entorno de simulación del robot en el software CoppeliaSim Edu. En este documento, se describe la conexión y la ejecución de los distintos programas necesarios para lograr una transferencia de datos exitosa entre Matlab/Simulink y CoppeliaSim Edu.

Los resultados obtenidos del controlador difuso son analizados y comparados con un controlador Proporcional Derivativo (PD) convencional, de manera que se pueda demostrar y evidenciar claramente las diferencias en el rendimiento del control y la precisión del seguimiento de las distintas trayectorias propuestas.

PALABRAS CLAVE: controlador, robot, inteligencia artificial, lógica difusa, programa, comunicación.

ABSTRACT

This paper presents the design and simulation of an artificial intelligence-based controller for the trajectory tracking of a humanoid robot NAO using fuzzy logic technique. The development of a high-level control system for the NAO robot is proposed, using a kinematic controller type PD based on fuzzy logic with trajectory compensation, and starting from the direct kinematic model of a mobile robot.

The controller is implemented in the Matlab/Simulink interface and a UDP communication is established with the simulation environment of the robot in CoppeliaSim Edu software. This document describes the connection and the execution of the various programs required to achieve a successful data transfer between Matlab/Simulink and CoppeliaSim Edu.

The results obtained from the fuzzy controller are analyzed and compared with a conventional Proportional Derivative (PD) controller, so that the differences in control performance and tracking accuracy of the different proposed trajectories can be clearly demonstrated and evidenced.

KEYWORDS: controller, robot, artificial intelligence, fuzzy logic, program, communication.

1 INTRODUCCIÓN

El mundo se encuentra nuevamente en una etapa de transición hacia una nueva revolución industrial: la era de la Inteligencia Artificial, transformando la realidad en la que se vive y, por ende, presentando nuevos desafíos [1].

La Inteligencia Artificial (IA) es una tecnología que estudia el desarrollo de algoritmos que imitan o emulan el pensamiento humano, realizando tareas como reconocer patrones, entender un lenguaje, tomar decisiones o aprender [2]. Algunos ejemplos más populares de IA son los asistentes de voz, como Siri y Alexa, los coches auto conducidos y las herramientas de software para reconocimiento de imágenes y texto. Actualmente, la IA también presenta múltiples aplicaciones en campos como la medicina, las finanzas, el control, la automatización y, en especial, la robótica [1], [2].

La robótica se ocupa del estudio, diseño y construcción de robots, máquinas mecánicas programadas para realizar tareas y desenvolverse en medio de determinados espacios y situaciones [1], [3], [4]. De ahí que el desarrollo de la inteligencia artificial y la robótica van de la mano con el objetivo de brindar a los robots una mayor capacidad para adaptarse a entornos complejos y variantes, comprender el ambiente en el que se encuentran y mejorar la interacción que tienen con los seres humanos [1].

Los robots pueden clasificarse de manera general en móviles, industriales y humanoides. Los robots móviles se desplazan mediante ruedas, patas u orugas en un entorno; los robots industriales manipulan, trasladan objetos y presentan una forma parecida a la de los brazos humanos; y los robots humanoides imitan parcial o totalmente el comportamiento humano [5]. El sistema de control de los robots permite que éstos ejecuten las tareas que se tenga planeadas para ellos, como control de posición y postura, seguimiento de trayectorias, entre otras. El diseño del controlador de un robot parte de un modelo matemático no lineal que represente el comportamiento del robot con gran exactitud [6].

Los robots humanoides requieren de piernas y brazos articulados similares a los humanos, que les permitan caminar, correr, manipular objetos, entre otros [7]. En consecuencia, la obtención de un modelo matemático que describa de manera exacta el comportamiento de un robot humanoide resulta muy compleja, lo que dificulta diseñar un controlador basándose en la teoría de control convencional [8]. Por esta razón, la investigación se ha centrado en determinar nuevas técnicas de control basadas en inteligencia artificial, que permitan obtener controladores para robots humanoides de manera más simple [6], [8].

NAO es uno de los robots humanoides programables más populares a nivel mundial, dado que presenta una estructura e interfaz muy amigable con el usuario. Fue diseñado exclusivamente para educación, investigación y asistencia médica [9].

El presente trabajo se enfocará en la técnica de inteligencia artificial denominada lógica difusa para controlar el seguimiento de trayectoria de un robot humanoide NAO. La lógica difusa permite establecer un sistema de control capaz de tomar decisiones en base a entradas y salidas con un determinado porcentaje de incertidumbre. Por tanto, no se requiere del modelo matemático del proceso a controlar (en este caso, el robot humanoide NAO) para diseñar el controlador [10].

En primer lugar, se presenta una revisión bibliográfica acerca de los conceptos teóricos de mayor relevancia que permiten la comprensión del contexto, procedimiento utilizado e interpretación de los resultados que se obtendrán del presente proyecto. Posteriormente, se detalla la metodología seguida para llevar a cabo el diseño del controlador basado en lógica difusa para el seguimiento de trayectorias de un robot humanoide NAO, el proceso que permite establecer la comunicación entre los programas Matlab/Simulink y CoppeliaSim Edu, y la generación de las trayectorias propuestas. Finalmente, se muestran los resultados obtenidos del controlador basado en lógica difusa ante las distintas trayectorias propuestas, se analiza su desempeño y se lo compara con un controlador tipo Proporcional Derivativo (PD) convencional.

1.1 OBJETIVO GENERAL

Diseñar y simular un controlador basado en lógica difusa para el seguimiento de trayectorias de un robot humanoide NAO.

1.2 OBJETIVOS ESPECÍFICOS

1. Realizar una revisión bibliográfica sobre los robots humanoides, el modelo cinemático de robots móviles, el control de seguimiento de trayectorias y sobre inteligencia artificial, haciendo énfasis en la técnica de lógica difusa.

2. Diseñar e implementar un controlador cinemático convencional y un controlador basado en lógica difusa que permitan realizar el seguimiento de trayectorias del robot NAO.

3. Establecer la comunicación entre los programas necesarios para la simulación del movimiento del robot NAO.

4. Probar el funcionamiento del controlador desarrollado, verificando que el robot NAO cumpla con el seguimiento de trayectorias.

5. Analizar el desempeño del controlador implementado mediante índices de desempeño utilizados en sistemas de control.

1.3 ALCANCE

- Se realizará una revisión bibliográfica sobre los robots humanoides haciendo énfasis en el robot humanoide NAO.

- Se realizará una revisión bibliográfica del modelo cinemático y control de seguimiento de trayectoria de robots móviles con el propósito de desarrollar un controlador basado en lógica difusa para el control de seguimiento de trayectoria del robot humanoide NAO.

- Se realizará una revisión bibliográfica acerca de los principales conceptos que conforman la base teórica de Lógica Difusa.

- Se diseñará e implementará un controlador tipo PD convencional en Matlab/Simulink que permita el control de seguimiento de trayectoria del robot humanoide NAO.

- Se diseñará e implementará un controlador difuso en Matlab/Simulink que permita el control de seguimiento de trayectoria del robot humanoide NAO.

- Se desarrollará el entorno de Simulación del robot humanoide NAO en el software CoppeliaSim Edu.

- Se realizarán pruebas de funcionamiento a nivel de simulación verificando que el robot NAO cumpla con el seguimiento de trayectorias.

- Se analizará el desempeño de los controladores mediante índices de desempeño utilizados en sistemas de control.

1.4 MARCO TEÓRICO

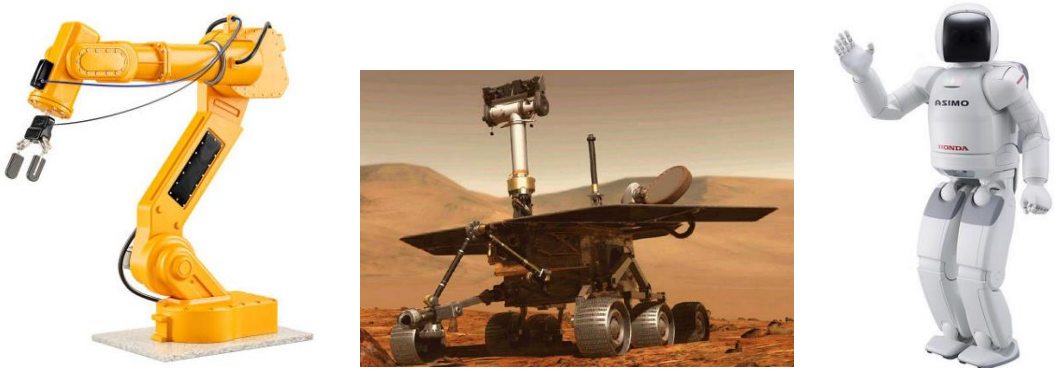
En esta sección, se muestra toda la revisión bibliográfica realizada con el objetivo de dar a entender el contexto teórico en el cual se basa el presente proyecto. De manera general, se aborda las características y especificaciones del robot humanoide NAO, los conceptos

de robótica, sistemas de control, inteligencia artificial y softwares necesarios para la implementación.

1.4.1 ROBÓTICA

La robótica se define como la ciencia que se encarga del estudio, diseño, construcción y operación de robots, combinando varios sistemas de naturaleza mecánica, electrónica e informática [1]. A lo largo de la historia, de acuerdo con las diferentes aplicaciones, clasificaciones y autores, los robots han ido adquiriendo múltiples interpretaciones; sin embargo, de manera general, se puede definir a los robots como máquinas provistas de sensores y actuadores, programadas para realizar tareas de manera autónoma o semiautónoma [5], [11].

Los robots se pueden clasificar, de acuerdo con sus funciones, en tres tipos principales: industriales, móviles y humanoides, como se observan en la Figura 1.1.



(a) (b) (c)

Figura 1.1. (a) Robot Manipulador Industrial [12]. (b) Robot Móvil Opportunity en Marte [13]. (c) Robot Humanoide Asimo [14].

1.4.1.1 Robots Industriales

Los robots industriales, también llamados manipuladores o brazos robóticos, son conformados por una cadena abierta de eslabones unidos mediante articulaciones [8]. Dicha cadena de eslabones posee un actuador final en un extremo y se encuentra anclada fijamente a una base en el otro extremo [11]. El actuador final cuenta con una herramienta especial que le permite realizar múltiples acciones, y su posición puede modificarse mediante el control del ángulo de giro de cada articulación que une cada eslabón [11], como se observa en la Figura 1.1a. La herramienta que se ubica en el actuador final permite al brazo robótico manipular, sujetar, soldar, pintar, ensamblar y muchas acciones más [6].

Por tal motivo, la aplicación de este tipo de robots es muy común en cadenas de fabricación en industrias [11].

1.4.1.2 Robots Móviles

Los robots móviles, como su nombre lo indica, tienen el objetivo de desplazarse por un entorno haciendo uso de ruedas, patas u orugas [11]. Los robots móviles que usan ruedas para trasladarse, también denominados vehículos autónomos, presentan diseño y construcción más simple; del mismo modo, su programación y control son más sencillos en comparación a otros mecanismos de locomoción [6]. Este tipo de robots son ampliamente utilizados para exploración científica en superficies interplanetarias (Figura 1.1b) y océanos, para la recolección de cultivos en la agricultura, en la extracción minera y muchas otras aplicaciones en las cuales el ser humano no puede acceder a ciertos espacios físicos o, simplemente, aplicaciones en las cuales la actividad humana puede ser reemplazada para un mejor rendimiento [5].

1.4.1.3 Robots Humanoides

Los robots humanoides tienen como objetivo imitar el comportamiento y actividades que los seres humanos realizan, como hablar, caminar, cantar, bailar, correr, saltar, transportar objetos, entre otras [7]. Su estructura física consta de un par de piernas, un par de brazos y una cabeza unidos a un tronco que, en conjunto, se asemeja a un cuerpo humano, como se observa en la Figura 1.1c.

Los robots humanoides son considerados una combinación entre los dos tipos de robot antes descritos [11]. Es decir, estos robots son capaces de realizar las tareas de ambos tipos: desplazarse por un entorno como los robots móviles y manipular objetos como los robots industriales [11]. Dado que el robot humanoide posee piernas como un ser humano, se lo puede considerar como robot móvil con locomoción de dos patas, dotado con dos brazos robóticos que le permiten sujetar y manipular objetos [6]. Es decir, se lo puede denominar manipulador móvil.

El objetivo de control más complejo de los robots humanoides es preservar la estabilidad estática y dinámica de toda la estructura física del robot, lo que significa que siempre debe mantenerse el equilibrio ante las distintas perturbaciones que éste experimente [6].

1.4.2 ROBOT NAO

NAO es un robot pequeño del tipo humanoide diseñado para interactuar con las personas. Se encuentra conformado por una serie de sensores y actuadores que le permiten caminar,

bailar, hablar y reconocer rostros y objetos. Fue creado en Francia por Aldebaran Robotic en el año 2008; sin embargo, en 2015, dicha empresa fue adquirida por SoftBank Robotics. NAO cuenta con un total de 6 generaciones en las cuales se destinó su uso alrededor del mundo para la investigación, educación y cuidado de la salud [15]. En la Figura 1.2 se muestra al robot NAO.

Figura 1.2. Robot Humanoide NAO [15].

1.4.2.1 Características Principales

Entre las características principales del robot humanoide NAO, en su sexta versión en el mercado (V6), se tiene que cuenta con 25 grados de libertad, un peso es de 5.5 [Kg], una altura de 58 [cm], longitud de 27.5 [cm] y profundidad de 31.1 [cm]. Puede alcanzar una velocidad máxima de caminata de 0.6 [Km/h], tiene una alimentación por batería de iones de litio de 27.6 [Wh], lo que representa una duración de 90 minutos de funcionamiento. Sus materiales de fabricación son el plástico de policarbonato-ABS, la poliamida y el termoplástico reforzado con fibra de carbono [15], [16].

El robot NAO cuenta con un procesador de cuatro núcleos Intel Atom a 1.91 [GHz], una memoria RAM de 4 [GB] DDR3 y 32 [GB] de almacenamiento en disco sólido SSD. Su sistema operativo es NAOqi, basado en Linux. Con respecto a las comunicaciones, este robot cuenta con sistemas de conexión Bluetooth, Wi-Fi y Ethernet. Para su programación y visualización dispone de una plataforma, denominada Choregraphe Suite, que es bastante amigable con el usuario [15], [16].

NAO tiene capacidad para rastrear objetos, reconocer voz y hablar hasta en 20 idiomas diferentes, entre los principales: inglés, español, francés, alemán, italiano, holandés,

portugués, finlandés, sueco, checo, árabe, ruso y turco. Además, presenta un gestor de recuperación de caídas [15], [16].

La razón principal para seleccionar al robot NAO como el robot humanoide para este proyecto radica en su sistema operativo, que facilita un control a alto nivel sobre su movimiento [17].

1.4.2.2 Sensores

A continuación, se enlistan los sensores con los que cuenta el robot humanoide NAO V6 para su funcionamiento e interacción con los seres humanos: dos cámaras OmniVision de 5 megapixeles, unidad inercial con acelerómetro de tres ejes y dos giroscopios, un telémetro sonar, cuatro micrófonos omnidireccionales, dos sensores infrarrojos, nueve sensores táctiles, ocho sensores de presión y veinticinco encoders magnéticos para medir la posición de las articulaciones del robot [15], [16].

1.4.2.3 Actuadores

El robot NAO V6 dispone de diversos actuadores, entre los cuales se incluyen varios LEDs ubicados en distintas partes del cuerpo, dos altavoces en la cabeza que le dan la capacidad de hablar y veinticinco motores Portescap de corriente continua sin escobillas [15]. Estos servomotores son responsables de las 25 articulaciones o grados de libertad (DoF) del robot, que están distribuidos de la siguiente manera: 2 DoF en la cabeza, 5 DoF en el brazo izquierdo, 5 DoF en el brazo derecho, 1 DoF en la pelvis, 5 DoF en la pierna izquierda, 5 DoF en la pierna derecha, 1 DoF en la mano izquierda y 1 DoF en la mano derecha [18], como se observa en la Figura 1.3.

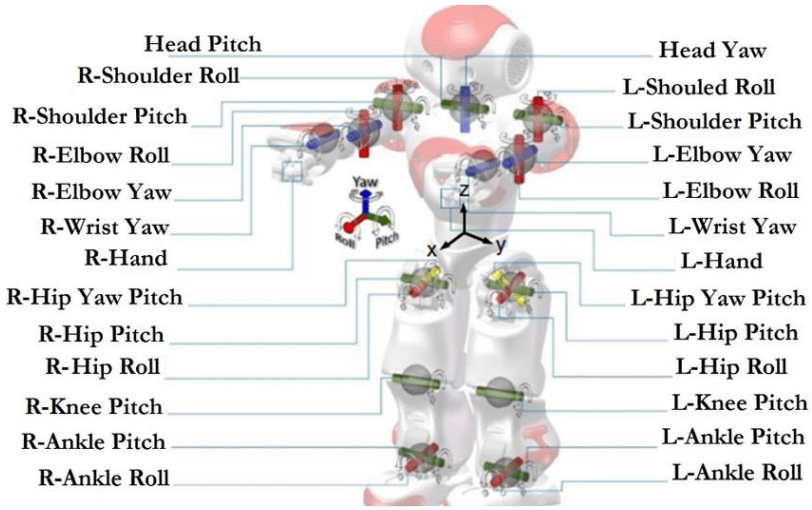


Figura 1.3. Articulaciones del robot NAO *[19]*.

1.4.3 SISTEMAS DE CONTROL EN ROBOTS

Un sistema de control puede definirse como un subsistema dentro de un sistema de mayor tamaño, planta, proceso o dispositivo, que es conformado por un conjunto de componentes que interactúan entre sí y trabajan juntos con el objetivo de modificar el comportamiento del sistema más grande (planta), para que éste funcione de una manera deseada [20], [21].

Un sistema de control automático presenta varios objetivos dependiendo de la aplicación, en el caso de procesos industriales, se busca optimizar el rendimiento y disminuir costos de producción [22]. En robótica, por definición, cada robot cuenta con un sistema de control autónomo, el cual tiene por objetivo la regulación y dirección de funciones que permitan al robot ejecutar los movimientos y cambios de estado deseados de manera precisa [23].

Los componentes de todo sistema de control son los siguientes [21]:

- La **planta** es aquel sistema del cual se obtiene una señal de salida, cuya magnitud se requiere controlar.
- La **señal de referencia** es la magnitud deseada que se busca en la señal de salida.
- El **sistema de sensores** permite la medición de la señal de salida.
- El **sistema de actuadores** posibilita la aplicación de acciones directas en la planta, con el propósito de alterar la señal de salida en función de la señal de control generada por el controlador.
- La **señal de error** es la diferencia entre la señal de referencia y la señal de salida, indica qué tan cerca se encuentran ambas señales.
- El **controlador** es el encargado de calcular la señal de control requerida en base a la señal de error obtenida. El objetivo del controlador es disminuir la señal de error a cero.
- Las **perturbaciones** son variaciones externas no deseadas que afectan al sistema de control tanto en su señal de salida como en la medición de los sensores.

Entre las características fundamentales que debe poseer cualquier controlador, se encuentra la robustez, lo que significa que debe ser capaz de adaptarse y mantener su eficiencia ante perturbaciones en el sistema. Además, un controlador debe ser estable y presentar un tiempo de ejecución de la acción de control pequeño [24].

Para el diseño de un sistema de control, se deben tener muy en cuenta las características y especificaciones de la planta, proceso, dispositivo o sistema en el cual se desea implementar, al igual que las variables o señales que se desean controlar. De esta manera,

el conocer un modelo matemático que permita describir el comportamiento de dicha planta brinda la capacidad de simular el sistema de control, diseñar controladores que se adapten genuinamente al sistema y realizar múltiples pruebas para evaluar su funcionamiento [22].

El sistema de control de los robots tiene múltiples aplicaciones, siendo las más utilizadas el control de postura y el control de seguimiento de trayectoria. El control de postura implica el adecuado control de cada articulación o rueda del robot para que éste pueda alcanzar una posición final deseada. Por otro lado, el control de seguimiento de trayectoria tiene como objetivo lograr que el robot se desplace a lo largo de una trayectoria determinada. Por lo que igual se requiere de un control preciso del ángulo de giro de las articulaciones en el caso de robots manipuladores y humanoides, así como de la velocidad de las ruedas en el caso de robots móviles.

Los robots humanoides presentan ciertos desafíos en términos de control, especialmente en lo que respecta a la robustez y la estabilidad. Estos inconvenientes están relacionados con la estructura física de los robots humanoides, ya que, al caminar, experimentan un balanceo natural debido a la posición de su centro de masa. Esto puede generar oscilaciones excesivas alrededor de la posición deseada, lo que dificulta mantener una respuesta estable y precisa.

1.4.4 CONTROL DE SEGUIMIENTO DE TRAYECTORIA

Uno de los principales objetivos de control en robótica es brindar un sistema de navegación apropiado al robot, que le permita desplazarse de una manera segura por un entorno de trabajo. Por tanto, es importante que el robot sea capaz de realizar seguimiento de trayectorias. En el caso del robot industrial, se requiere que el actuador final cambie su posición en el espacio, de manera que logre seguir una trayectoria. Por otro lado, en robots móviles y humanoides, se busca que la posición del centro de masa de éstos varíe para que se mantenga dicha trayectoria deseada [25].

Una trayectoria es descrita como el conjunto de posiciones (o camino) que ocupa un objeto mientras se mueve en un determinado tiempo [26]. En robótica, se requiere que un robot recorra cada elemento del conjunto de posiciones deseadas en un determinado instante de tiempo. A esto, se le denomina seguir la trayectoria deseada [27]. Por tanto, es importante que el entorno de trabajo sea conocido y se encuentre libre de obstáculos que puedan interferir en el libre movimiento del robot a lo largo de aquella trayectoria deseada [27].

Para el diseño del controlador de seguimiento de trayectoria, se debe conocer el modelo matemático del robot que relacione la entrada de control y el comportamiento del sistema [25]. En el caso del robot móvil, gracias a sus ruedas, el modelo matemático es muy sencillo de obtener; sin embargo, en el caso del robot humanoide, esta tarea puede ser muy compleja. Se debe recordar que, al presentar piernas, es normal que el robot humanoide presente cierto balanceo al momento de caminar a lo largo de una trayectoria, lo que se traduce en no linealidades que complican el modelamiento preciso del robot.

1.4.5 ÍNDICES DE DESEMPEÑO DE CONTROL

Los índices de desempeño son medidas cuantitativas que permiten evaluar la eficiencia que presenta un sistema de control. De manera general, los índices de desempeño se obtienen aplicando una sumatoria acumulada de los parámetros del sistema, como errores, señales de control y tiempo. Se considera que el sistema tiene un comportamiento óptimo (o eficiente) cuando estos indicadores alcanzan un valor mínimo [28]. De esta forma, se puede comparar el rendimiento de distintos controladores obteniendo sus índices de desempeño. A continuación, se definen los índices de desempeño más utilizados en sistemas de control.

ISE: Este indicador brinda una medida del valor del error acumulado del sistema. Es decir, permite conocer qué tan cerca estuvo la señal controlada de la señal de referencia a lo largo del tiempo [28].

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

ISU: Este indicador brinda una medida del valor de la acción de control acumulada del sistema. Es decir, permite conocer la energía empleada por el controlador (o esfuerzo de control) a lo largo del tiempo [29].

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

TVu: Este indicador brinda una medida del valor de las variaciones de acción de control acumuladas del sistema. Es decir, permite conocer la evolución de la acción de control a lo largo del tiempo [29]. Si el valor de TVu es elevado significa que la señal de control varía bruscamente; al contrario, si el TVu es bajo se puede decir que la señal de control es suave.

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

1.4.6 INTELIGENCIA ARTIFICIAL

La Inteligencia Artificial (IA) es un campo de la informática que se dedica al desarrollo de sistemas y tecnologías capaces de llevar a cabo tareas que normalmente requieren inteligencia humana. Estas tareas incluyen el aprendizaje, la percepción, el razonamiento resolución de problemas y la toma de decisiones. Para lograrlo, la IA se apoya en algoritmos y modelos matemáticos complejos que permiten a las máquinas procesar grandes volúmenes de datos y aprender de ellos para mejorar su rendimiento. Con aplicaciones que abarcan desde asistentes virtuales hasta robots de diversa índole, la IA está transformando rápidamente tanto la forma de trabajar como de vivir de la humanidad [1].

La inteligencia artificial presenta múltiples campos en los cuales basa su funcionamiento. Estos campos, también conocidos como enfoques o técnicas, son utilizados para la resolución de distintos problemas en varias aplicaciones dentro de la inteligencia artificial. Entre las técnicas de IA más básicas que existen, y a partir de las cuales se han desarrollado muchas más, se tiene a la lógica difusa, las redes neuronales y los algoritmos genéticos [30].

1.4.6.1 Sistemas de Lógica Difusa

La lógica difusa es un método matemático que se utiliza para el estudio del razonamiento y la toma de decisiones basados en información imprecisa o incierta. Se fundamenta en la teoría de conjuntos difusos, la cual permite establecer o determinar el grado de pertenencia parcial que un elemento tiene respecto a un conjunto. La lógica difusa encuentra una amplia gama de aplicaciones en áreas como el reconocimiento de patrones, los sistemas de control y la toma de decisiones [10], [30].

Los sistemas de lógica difusa son una técnica de inteligencia artificial que permite manejar de manera simultánea datos numéricos, información conceptual y conocimientos empíricos humanos, expresados en lenguaje natural con el objetivo de tomar decisiones en el sistema. El hecho de incorporar expresiones lingüísticas los hace especialmente útiles en situaciones donde no se disponen de mediciones precisas, o cuando éstas no son prácticas de obtener [10], [31]. En el presente proyecto se realizará un estudio de la lógica difusa enfocada hacia sistemas de control.

1.4.7 SOFTWARE DE SIMULACIÓN

A continuación, se describen los softwares de simulación que son necesarios para que la simulación del control de seguimiento de trayectoria del robot humanoide NAO se lleve a

cabo. Para ello, es necesario que se establezca una conexión entre los distintos programas involucrados.

1.4.7.1 Matlab/Simulink

Simulink es un software de simulación que presenta un entorno de diagramas de bloque muy amigable con los usuarios. Su interfaz es muy intuitiva y simple de comprender, no requiere de programación escrita para su funcionamiento; sin embargo, puede ser utilizado en conjunto con Matlab para optimizar el ingreso de parámetros importantes de las simulaciones y para analizar los resultados de éstas [32]. El funcionamiento de Simulink se basa en la interconexión de bloques de funciones capaces de realizar distintas operaciones y tareas, como la configuración y simulación de modelos matemáticos [32].

Entre los principales bloques de Simulink que se utilizarán en el presente proyecto, se tiene el “UDP Send” y “UDP Receive”, que permiten la comunicación mediante protocolo UDP, como se observa en la Figura 1.4 [33].

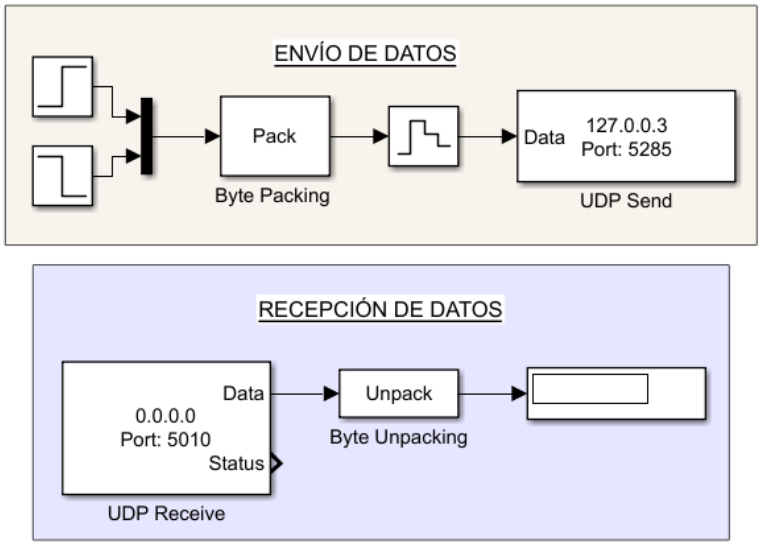


Figura 1.4. Diagrama de bloques para la comunicación UDP en Simulink.

1.4.7.2 CoppeliaSim Edu

CoppeliaSim es un software de simulación de robótica que presenta un entorno de desarrollo integrado y proporciona un gran número de herramientas que permiten la simulación de distintos tipos de robots [34], [35]. Este software es capaz de diseñar el modelo virtual tridimensional de cualquier objeto o robot y simular su funcionamiento en tiempo real [35]. CoppeliaSim emplea una arquitectura de control distribuido que brinda la capacidad al usuario para controlar cada objeto o modelo de manera individual, incluso se pueden desarrollar controladores en Python, Matlab/Simulink, C/C++ o Java, e integrarse

mediante plug-ins, nodos ROS o clientes API remotos [34]. Esto lo convierte en un software muy versátil que permite considerar múltiples características cinemáticas y dinámicas de cada robot al momento de simular. En la Figura 1.5, se puede observar la interfaz de CoppeliaSim Edu.

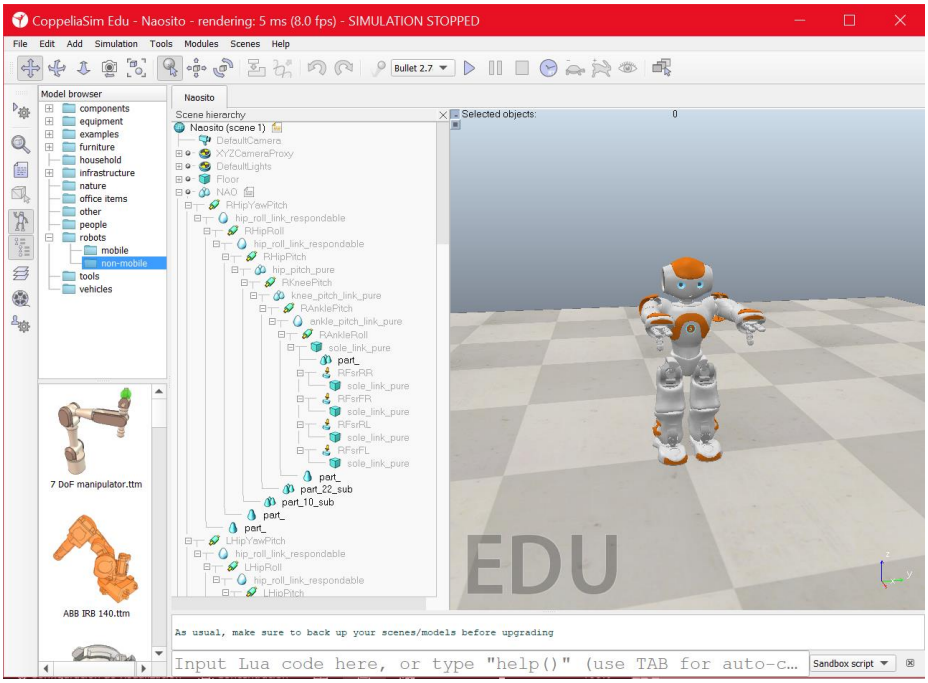


Figura 1.5. Entorno de Simulación CoppeliaSim Edu.

1.4.7.3 NAOqi

NAOqi es un framework que brinda el acceso a todas las funcionalidades del robot humanoide NAO, tanto a bajo como a alto nivel. Las funciones a bajo nivel que se pueden controlar con NAOqi son la lectura de datos de los sensores, envío de comandos de posición angular hacia los servomotores, encendido y apagado de LEDs, entre otros. Por otro lado, las funciones a alto nivel son comandos más estructurados que le permiten al robot levantarse, caminar, hablar, bailar, girar, entre otros. Todas aquellas funcionalidades y comportamientos mencionados del robot pueden ser programados en C++, Python y Urbi [36]. NAOqi se ejecuta en Linux en el CPU del robot NAO, y en cualquier sistema operativo en PC o Mac. Con esto, se proporciona a los usuarios un amplio abanico de posibilidades para el control del funcionamiento del robot, desarrollando y ejecutando scripts en los lenguajes de programación mencionados [36], [37]. Es importante mencionar que NAOqi actualmente solo puede interactuar con la versión 2.7 de Python [38].

1.4.7.3.1 Choregraphe Suite

Choregraphe Suite es un software que permite la programación de comportamientos del robot NAO mediante bloques de función en una interfaz muy amigable con el usuario, como se aprecia en la Figura 1.6. Este software es un módulo específico de NAOqi que se ejecuta en un ordenador. Es decir, NAOqi se ejecuta detrás de Choregraphe, facultando que éste acceda a las funcionalidades del robot NAO.

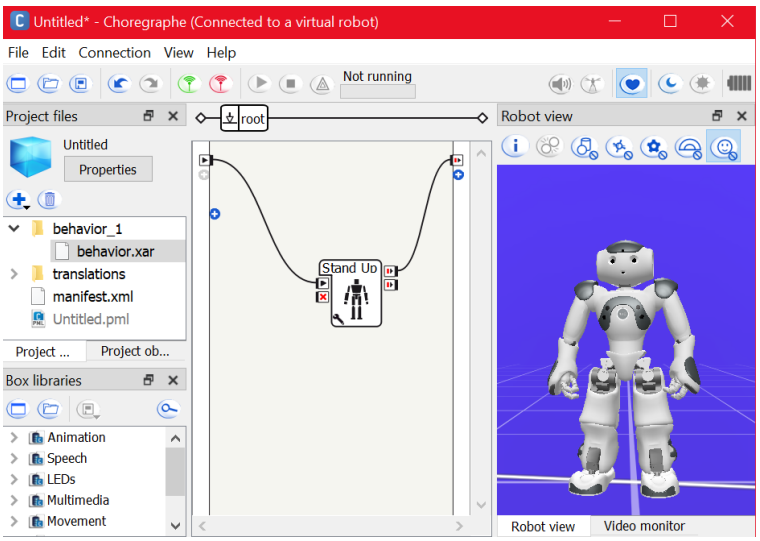


Figura 1.6. Interfaz de Choregraphe Suite.

1.4.7.4 Python

Python es uno de los lenguajes de programación más utilizados actualmente debido a su elevada aplicación en desarrollo web, desarrollo de videojuegos, aplicaciones móviles, ciencia de datos e Inteligencia Artificial. Fue creado en 1990 por Guido Van Rossum, basándose en el lenguaje de programación a alto nivel C. Python es un lenguaje muy versátil que presenta múltiples librerías y extensiones que permiten su compatibilidad con muchos otros lenguajes de programación [39].

1.4.7.4.1 Spyder

Spyder es un entorno de desarrollo integrado (IDE) de código abierto que permite la programación en Python. Este IDE fue desarrollado para el uso científico, en ingeniería y análisis de datos, contando con múltiples herramientas y librerías para la edición, análisis y depuración de código escrito en Python [40].

2 METODOLOGÍA

En el presente proyecto, se propone el desarrollo de un controlador difuso para el seguimiento de trayectorias de un robot humanoide NAO, partiendo del modelo cinemático de un robot móvil. Para su desarrollo, es necesario el uso de los siguientes softwares: Matlab/Simulink, donde se programan las trayectorias deseadas y el controlador propuesto; Spyder, que permite la programación en Python para la comunicación; Choregraphe Suite, que ejecuta el framework NAOqi para el control del robot NAO; y CoppeliaSim Edu, donde se simula un robot NAO en medio de un entorno de trabajo.

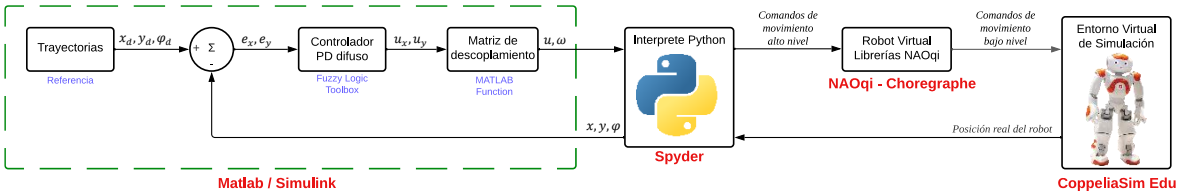


Figura 2.1. Diagrama de bloques funcional: Simulación y Control del robot NAO.

Dentro de Matlab/Simulink, se programan las distintas trayectorias que le servirán al robot como referencia de las posiciones deseadas y el controlador propuesto. Se hace uso del "Fuzzy Logic Toolbox" de Matlab para la programación e implementación del controlador difuso en Simulink. Además, se utiliza una función de Matlab para ejecutar la matriz de desacoplamiento que permite obtener las señales de control que comandarán el movimiento del robot NAO.

Posteriormente, en el entorno de desarrollo de Python, Spyder, se ejecutan los comandos que permiten la comunicación entre el controlador desarrollado, el robot virtual y su entorno de simulación. Aquí, se programa el envío de las señales de control en alto nivel hacia el NAOqi (robot virtual) y la recepción de la posición real del robot NAO desde el entorno de simulación CoppeliaSim Edu.

Al iniciar el software Choregraphe, se activa NAOqi, el framework encargado de enviar comandos de movimiento a un robot NAO virtual. A través de la interfaz gráfica de Choregraphe, se puede observar el movimiento del robot virtual. Partiendo de las señales de velocidad lineal y angular, NAOqi envía los comandos necesarios al robot NAO del entorno de simulación para controlar el ángulo de giro de cada articulación de sus piernas.

Finalmente, CoppeliaSim Edu se enlaza con NAOqi mediante Python para obtener los valores de las posiciones angulares de cada articulación, mismos que permiten que el robot NAO camine. A la vez, CoppeliaSim Edu envía los datos de posición del robot hacia Spyder.

Para analizar el funcionamiento del sistema de control, se extraen los resultados del controlador difuso en forma de gráficas e índices de desempeño. Adicionalmente, se diseña un controlador PD convencional con el objetivo de comparar el enfoque del controlador propuesto con el enfoque del controlador tradicional.

2.1 MODELO DEL ROBOT HUMANOIDE NAO

Un modelo matemático relaciona la entrada de control con el comportamiento de un sistema [22]. En el caso de un robot, cuando éste se mueve a bajas velocidades, el modelo cinemático describe perfectamente su comportamiento; por otro lado, a altas velocidades, el robot empieza a experimentar ciertas fuerzas debido al movimiento, por tanto, el modelo dinámico describirá mejor su comportamiento en comparación con modelo cinemático [35].

Como se había mencionado en la sección 1.4.2, en el presente enfoque se escoge el robot humanoide NAO debido a las facilidades que presenta para realizar el control de movimiento a alto nivel. Gracias a NAOqi y, en especial, al API “ALMotion”, se puede llevar a cabo el movimiento de caminata del robot NAO partiendo de señales de velocidad lineal y velocidad angular [41]. Es decir, el sistema operativo del robot NAO cuenta con un control de articulaciones que le brinda la capacidad de caminar sobre una superficie plana a velocidades de movimiento deseadas.

De acuerdo con [42] y [43], cuando el terreno de caminata es plano, es posible aproximar el modelo de un robot humanoide al de un robot móvil.

En base a lo mencionado y a que el robot humanoide NAO que se plantea controlar no es capaz estructuralmente de alcanzar elevadas velocidades al momento de caminar, se considera que el modelo matemático que describe su comportamiento es el modelo cinemático de un robot móvil diferencial. Es decir, en base a las condiciones de funcionamiento, el modelo del robot humanoide es aproximado al modelo de un robot móvil a manera de hallar una relación entre las velocidades de control, posición y orientación del robot NAO.

2.1.1 CINEMÁTICA DEL ROBOT MOVIL

En la Figura 2.2 se muestra un robot móvil diferencial cuyo centro geométrico (punto medio del eje que une las ruedas) y centro de masa se encuentran en posiciones distintas, como es usual en este tipo de robots [25].

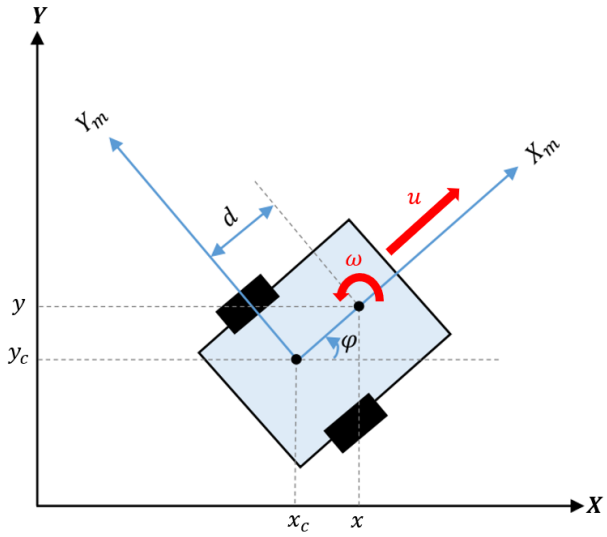


Figura 2.2. Cinemática del robot móvil diferencial *[25]*.

Se observa que:

- (x_c, y_c) es la posición del centro geométrico.
- (x, y) es la posición del centro de masa que se desea controlar.
- φ es la orientación del robot.
- d es la distancia entre los centros de masa y geométrico.
- u es la velocidad lineal del robot.
- ω es la velocidad angular del robot.

El modelo cinemático completo del robot móvil diferencial se encuentra expresado en las ecuaciones (2.1), (2.2) y (2.3) [25].

$$\dot{x}(t) = u(t)\cos\big(\varphi(t)\big) - d\omega(t)\sin\big(\varphi(t)\big). \tag{2.1}$$

$$\dot{y}(t) = u(t)\sin\big(\varphi(t)\big) - d\omega(t)\cos\big(\varphi(t)\big). \tag{2.2}$$

$$\dot{\varphi}(t) = \omega(t). \tag{2.3}$$

Expresando el modelo de manera matricial, se tiene:

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

Se puede verificar que el modelo cinemático muestra la relación entre las velocidades de movimiento en el eje X y eje Y con las velocidades lineal y angular del robot.

Se puede definir el vector de velocidades de movimiento del robot como:

$$\dot{h} = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix}. \tag{2.5}$$

Al integrar (2.5), se obtiene el vector de posición del robot, como se observa en la ecuación (2.6).

$$h = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}. \tag{2.6}$$

El vector de las velocidades de control del robot se expresa en (2.7).

$$U = \begin{bmatrix} u(t) \\ \omega(t) \end{bmatrix}. \tag{2.7}$$

Luego, reescribiendo el modelo cinemático del robot móvil, se obtiene

$$\dot{h} = JU, \tag{2.8}$$

donde J es la matriz Jacobiana del modelo del robot.

$$J = \begin{bmatrix} \cos(\varphi(t)) & -d\sin(\varphi(t)) \\ \sin(\varphi(t)) & d\cos(\varphi(t)) \end{bmatrix}. \tag{2.9}$$

Partiendo del modelo matemático expresado en la ecuación (2.8), se realizará el diseño de los controladores planteados.

2.2 LÓGICA DIFUSA

Lofty Aliasker Zadeh, profesor de la Universidad de Berkeley (California), desarrolló el concepto de lógica difusa en 1965 [22], [44]. Este concepto hace referencia a las ambigüedades que surgen cuando las personas evalúan su entorno. Es decir, cada individuo puede percibir e interpretar el ambiente que lo rodea de manera diferente. Por ejemplo, la velocidad de un automóvil viajando a 50 [Km/h] puede ser descrita por una persona como rápida, por otra como lenta, y por otra como media [45]. De igual forma sucede con muchas otras variables observables, como la altura de una persona o la temperatura dentro de casa. Partiendo de esto, la lógica difusa establece toda una base teórica de conjuntos, que permiten definir dichas ambigüedades y realizar operaciones entre éstos a manera de obtener otro conjunto difuso de salida [45].

A mediados de los 70 fue cuando la lógica difusa empezó a aplicarse en sistemas de control, dando lugar a los sistemas de inferencia difusos [22]. Estos sistemas se basan en el establecimiento de sentencias lógicas que permiten emular el pensamiento humano, lo que fundamenta la base para convertirse en una forma de inteligencia artificial [45].

La lógica difusa tiene su principal centro de desarrollo en Japón, donde sus investigadores la han aplicado principalmente en sistemas de control de electrodomésticos [44]. Sin embargo, sus aplicaciones han ido en aumento y en la actualidad se combina su funcionalidad con otras técnicas de control o inteligencia artificial para potenciar su utilidad [45].

Dentro de la teoría de lógica difusa, se deben analizar los conceptos de conjuntos difusos, funciones de membresía, valores lingüísticos, universo de discurso y operaciones difusas.

2.2.1 CONJUNTOS DIFUSOS

En la Figura 2.3a se observa un conjunto A convencional, cuyo límite está claramente definido, lo que permite concluir que los elementos que se encuentran dentro del límite, como x_2, pertenecen a A, mientras que aquellos que están fuera del límite, como x_1, no pertenecen a dicho conjunto. Por otro lado, en la Figura 2.3b, se aprecia un conjunto A cuyo límite no se encuentra definido claramente, por lo que la pertenencia o no de cada elemento dependerá de cada observador, éste es un conjunto difuso. A simple vista, se puede decir que x_2 pertenece a A; sin embargo, de x_1, se puede concluir que presenta un grado de pertenencia parcial a dicho conjunto [45].

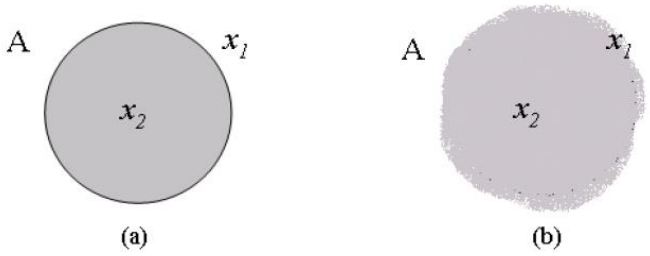


Figura 2.3. (a) Conjunto convencional. (b) Conjunto difuso *[45]*.

Por tanto, un conjunto difuso está compuesto por elementos que tienen distinto grado de pertenencia a dicho conjunto (pertenencia parcial). Es decir, existen algunos elementos que presentan un mayor grado de pertenencia que otros. Además, es importante destacar que los mismos elementos con menor grado de pertenencia en un conjunto difuso específico pueden presentar otro grado de pertenencia en otros conjuntos difusos dentro del conjunto universo [22], [45].

2.2.2 FUNCIONES DE MEMBRESÍA

Una función de membresía es la función característica de un conjunto difuso y representa el grado de pertenencia de un elemento a dicho conjunto difuso [22]. Estas funciones toman valores en el rango de 0 a 1. Un valor de 1 indica una pertenencia completa al conjunto difuso, mientras que un valor de 0 indica no pertenencia. Además, estas funciones pueden tener distintas formas o tipos, como triangular, sigmoidal, trapezoidal o campana de Gauss [44]. Cada forma puede variar de acuerdo con los requerimientos de control y el diseñador.

A continuación, en la Figura 2.4, se observa la función de membresía de un conjunto convencional, es decir, aquel en el cual sus elementos únicamente presentan un grado de pertenencia de 1 ó 0 [22]. En este caso, la única forma que define a esta función de pertenencia es la Singleton [44].

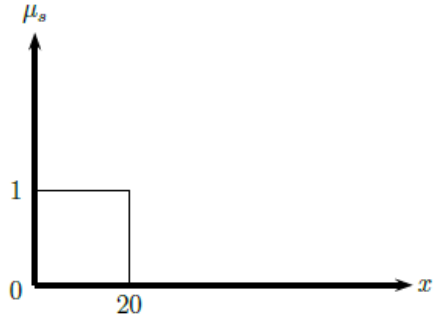


Figura 2.4. Función de membresía de un conjunto convencional *[22]*.

Luego, en la Figura 2.5 se observa las funciones de membresía de distintos conjuntos difusos.

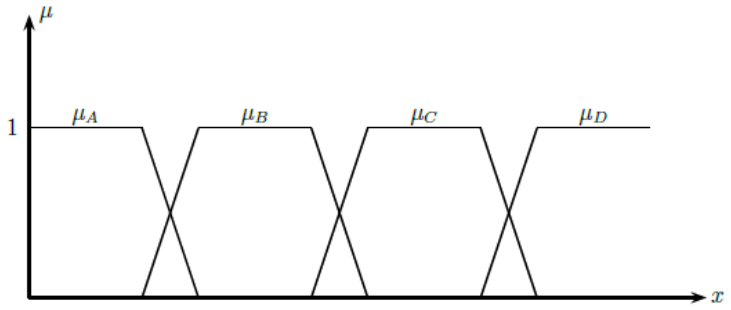


Figura 2.5. Funciones de membresía de conjuntos difusos *[22]*.

La función de membresía utilizada en la Figura 2.5 para cada conjunto difuso (μ_A, μ_B, μ_C, μ_D) es de tipo trapezoidal [22].

2.2.3 VALORES LINGUÍSTICOS

Un valor lingüístico es el nombre o variable lingüística que se le asocia a cada conjunto difuso en base al valor de una variable, como se observa en la Figura 2.6, donde se utiliza el ejemplo de la temperatura [22].

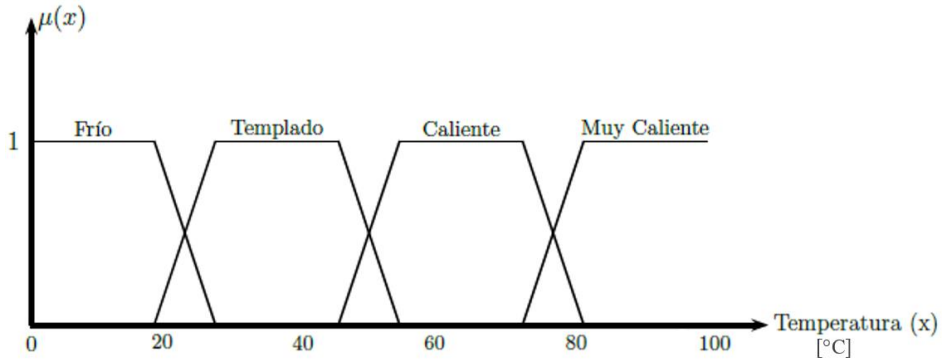


Figura 2.6. Variables Lingüísticas asociadas a conjuntos difusos *[22].*

La asignación de variables lingüísticas se la realiza en base a los conocimientos que se tenga de los elementos del conjunto difuso, en este caso, los valores de temperatura [22].

2.2.4 UNIVERSO DEL DISCURSO

El universo de discurso se define como el conjunto universo que se subdivide en varios conjuntos difusos. Es decir, es el conjunto de valores que una variable puede tomar. Por ejemplo, en la Figura 2.6, se puede observar que el universo de discurso para aquel caso

se encuentra definido entre 0 y 100 °C. Por tanto, se puede concluir que el universo del discurso es el dominio de las funciones de membresía de todos los conjuntos difusos [45].

2.2.5 OPERACIONES DIFUSAS

Las operaciones difusas son aquellas operaciones que se realizan entre conjuntos difusos, y que permiten obtener otros conjuntos difusos. Existen tres operaciones básicas en lógica difusa [44], cuya representación gráfica se muestra en la Figura 2.7.

- Complemento.

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{2.10}$$

- Unión u operador lógico OR.

$$\mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)] \tag{2.11}$$

- Intersección u operador lógico AND.

$$\mu_{A \cap B}(x) = \min[\mu_A(x), \mu_B(x)] \tag{2.12}$$

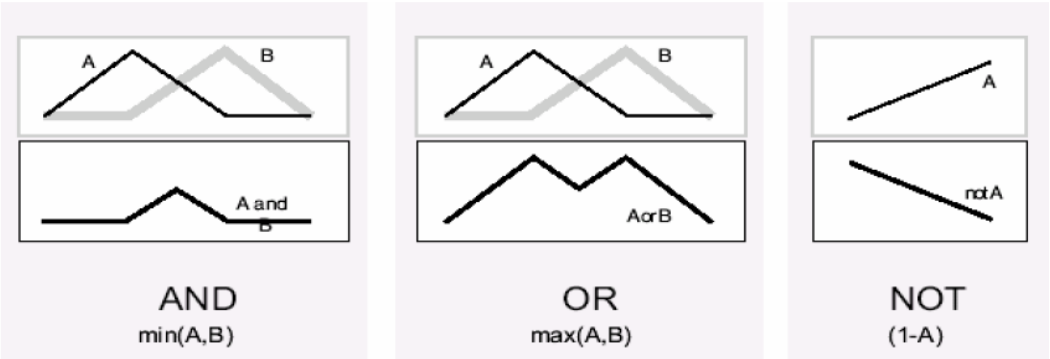


Figura 2.7. Operadores difusos *[44]*.

2.3 CONTROL DIFUSO

El objetivo del sistema de control difuso es determinar lógicamente la manera en la que debe actuar el sistema para alcanzar los objetivos de control requeridos, a partir de una base de conocimientos concedidos por un operador humano. La experiencia y el conocimiento humano proporcionado ayuda al sistema a tomar las acciones y decisiones correspondientes para realizar el control adecuadamente [22], [44].

Entre las ventajas con las que cuenta la lógica difusa, se tiene que no se requiere del modelo matemático del proceso a controlar y el controlador se obtiene empíricamente sin complicaciones matemáticas [22], [44].

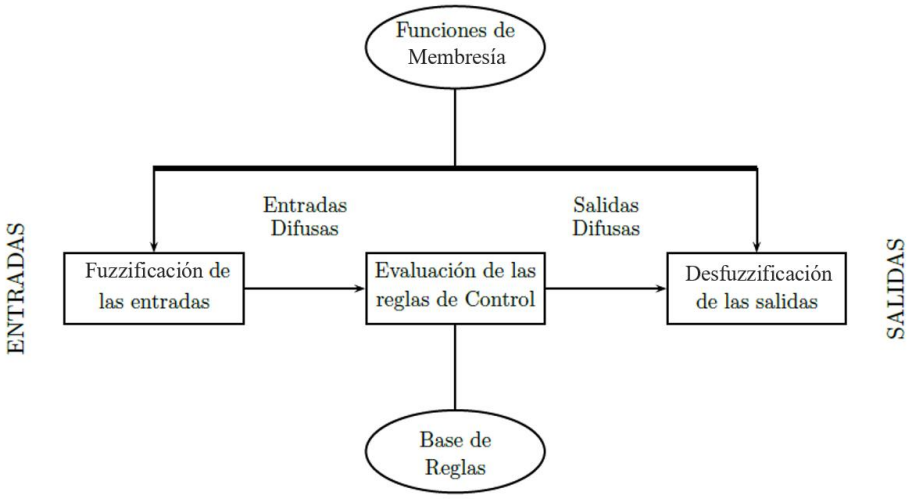


Figura 2.8. Estructura del sistema de inferencia difuso / controlador difuso *[22]*.

El control difuso se lleva a cabo mediante el proceso específico que se muestra en el esquema de la Figura 2.8. Dicho proceso se encuentra respaldado por toda la base teórica de lógica difusa anteriormente presentada [22].

2.3.1 DEFINICIÓN DE FUNCIONES DE MEMBRESÍA

Como se había mencionado, las funciones de membresía son aquellas que definen el grado de pertenencia de una variable (o elemento) a un conjunto difuso. En el sistema de inferencia difuso es necesario definir un grupo de funciones de membresía para cada variable de entrada y salida que exista en el controlador. Por ejemplo, si el sistema cuenta con 2 entradas y una salida, se deben definir 3 grupos de funciones de membresía para los conjuntos difusos de cada variable. Cada conjunto difuso tiene asociada su respectiva variable lingüística que, junto con su función de pertenencia, son fundamentales para la inferencia y el proceso de control difuso. Esto posibilita expresar el conocimiento lingüístico y transformarlo en reglas de control que guíen el comportamiento del sistema.

2.3.2 FUZZIFICACIÓN

Esta etapa consiste en la transformación de las variables medidas de entrada (elementos del universo del discurso) en un valor de grados de pertenencia a cada conjunto difuso, basándose en las funciones de membresía correspondientes. Es decir, este proceso devuelve múltiples valores entre 0 y 1 a partir de la variable de entrada. El número de

valores que devuelve este proceso depende de la cantidad de conjuntos difusos que contenga dicha variable de entrada [44].

2.3.3 BASE DE REGLAS

El control difuso requiere de una base de conocimiento para poder tomar las acciones o decisiones de control pertinentes. Esta base de conocimiento se expresa como una base de reglas que es proporcionada partiendo de la experiencia y conocimiento del operador humano. Es aquí donde el controlador difuso basa su funcionamiento [44], [45].

Una regla difusa presenta una estructura SI – ENTONCES, donde el argumento que se encuentra después de SI se denomina antecedente o premisa y el argumento que va después de ENTONCES es el consecuente o consecuencia. En un antecedente, se asocia una variable de entrada con una variable lingüística (conjunto difuso). Se puede utilizar varios antecedentes en una regla haciendo uso de operadores difusos AND, OR y NOT. Por otro lado, en el consecuente, se asigna una variable lingüística (conjunto difuso) a la variable de salida [44].

Existen dos tipos principales de reglas, Mamdani y Takagi – Sugeno, de los cuales el primero es el más utilizado [22], [44].

2.3.4 EVALUACIÓN DE LAS REGLAS DE CONTROL

Este proceso se encuentra conformado por tres subetapas: evaluación de reglas difusas, inferencia y agregado.

Evaluación de las reglas difusas: Se realizan las operaciones difusas que dicta cada regla de la base de reglas, partiendo de los valores obtenidos del proceso de fuzzificación.

Inferencia: En esta subetapa, se realiza el cálculo del conjunto difuso de salida de cada regla en base a las respectivas funciones de membresía de la variable de salida. Existen cuatro métodos principales de inferencia: Mamdani, Larsen, Drastic y Bounded. El método más usado es el de Mamdani, que corresponde a la aplicación del operador de valor mínimo. Después de aplicar el método de inferencia deseado, se obtienen tantos conjuntos difusos de salida como reglas existan [44].

Agregado: En esta subetapa, se realiza la agrupación de los múltiples conjuntos difusos de salida obtenidos en la anterior etapa de inferencia. De esta manera, se produce el conjunto difuso de salida total. Existen diversos métodos para la agrupación; sin embargo, el más utilizado es la aplicación de la operación del valor máximo [44].

2.3.5 DESFUZZIFICACIÓN

Esta etapa se encarga de realizar el proceso contrario a la fuzzificación, es decir, transforma el conjunto difuso de salida obtenido en la anterior etapa en un valor real a la salida. Existen múltiples métodos de desfuzzificación: centro de gravedad (COG), centro de área (COA), criterio máximo (MC), media del máximo (MOM), máximo más pequeño (SOM), máximo más grande (LOM) y bisector de área (BOA). De todos los métodos mencionados, el más utilizado es el método del centro de área [44].

2.4 DISEÑO DE TRAYECTORIAS DE REFERENCIA

Una trayectoria se encuentra definida, de acuerdo con la ecuación (2.13), por las componentes de movimiento (X y Y) parametrizadas en el tiempo. En el presente proyecto se usan tres trayectorias de referencia diferentes, como se observa en la Figura 2.9.

$$h_d = \begin{bmatrix} x_d(t) \\ y_d(t) \end{bmatrix}. \tag{2.13}$$

La trayectoria circular está definida por la ecuación (2.14).

$$\begin{cases} x_d(t) = \cos\left(\dfrac{t}{450}\right) \\ y_d(t) = \sin\left(\dfrac{t}{450}\right) \end{cases} \tag{2.14}$$

La trayectoria lemniscata está definida por la ecuación (2.15).

$$\begin{cases} x_d(t) = \sin\left(\dfrac{t}{450}\right) \\ y_d(t) = \sin\left(\dfrac{t}{225}\right) \end{cases} \tag{2.15}$$

La trayectoria cuadrada está definida por la ecuación (2.16).

$$\begin{cases} x_d(t) = \begin{cases} 1 - \dfrac{t}{200}, & 0 \le t < 400 \\ -1, & 400 \le t < 800 \\ -5 + \dfrac{t}{200}, & 800 \le t < 1200 \\ 1, & 1200 \le t < 1600 \end{cases} \\ y_d(t) = \begin{cases} 1 & 0 \le t < 400 \\ 3 - \dfrac{t}{200} & 400 \le t < 800 \\ -1 & 800 \le t < 1200 \\ -7 + \dfrac{t}{200} & 1200 \le t < 1600 \end{cases} \end{cases} \tag{2.16}$$

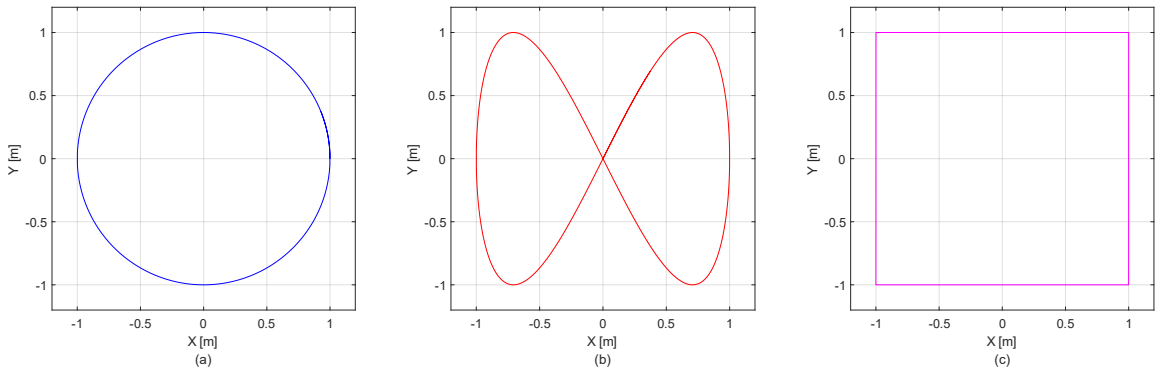


Figura 2.9. Trayectorias de referencia. (a) Circular. (b) Lemniscata. (c) Cuadrada.

2.5 DISEÑO DEL CONTROLADOR PD CONVENCIONAL

En la sección 2.1.1 se estudia la cinemática del robot móvil diferencial, cuyo modelo se utilizará para el diseño de un controlador de seguimiento de trayectoria de un robot humanoide NAO.

Partiendo de la ecuación (2.8), se tiene

$$U = J^{-1}\dot{h}. \tag{2.17}$$

Donde J^{-1} es la matriz de desacoplamiento. Luego,

$$U = J^{-1}u_c. \tag{2.18}$$

Se propone un controlador cinemático tipo Proporcional Derivativo (PD) con compensación de trayectoria, cuya ley de control se encuentra definida en la ecuación (2.19).

$$\begin{aligned} u_c &= \dot{h_d} + K_p e + K_d \dot{e}; \\ K_p &= k_p \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, k_p \in \mathbb{R}^+. \\ K_d &= k_d \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, k_d \in \mathbb{R}^+. \end{aligned} \tag{2.19}$$

La compensación viene dada por la derivada de la trayectoria,

$$\dot{h_d} = \begin{bmatrix} \dot{x_d}(t) \\ \dot{y_d}(t) \end{bmatrix}. \tag{2.20}$$

En la ecuación (2.20), $\dot{x_d}(t)$ y $\dot{y_d}(t)$ son las velocidades deseadas de las componentes de movimiento en cada instante de tiempo. Además, el error se expresa:

$$e = h_d - h. \tag{2.21}$$

Las constantes k_p y k_d del controlador PD son sintonizadas de manera heurística.

Como se puede verificar en (2.19), las entradas del controlador son el error de posición, e, su derivada, $\dot{e}$, y la compensación de trayectoria, $\dot{h_d}$; por otro lado, su salida es la acción de control, u_c, como se observa en el diagrama de bloques de la Figura 2.10.

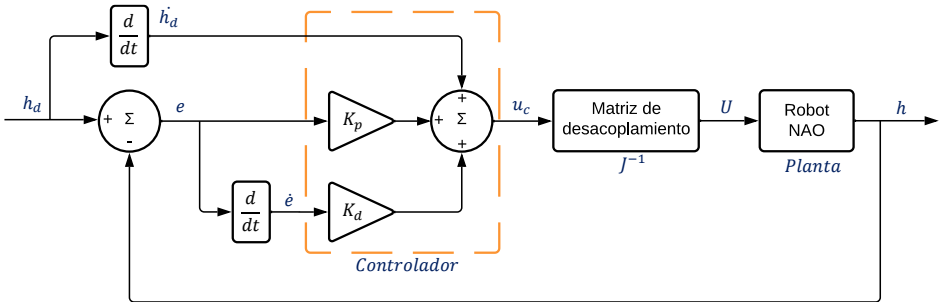


Figura 2.10. Diagrama de bloques del sistema de control del robot NAO usando un controlador PD convencional.

El objetivo del controlador es hacer que el robot reduzca siempre el error entre la posición deseada y su posición actual.

2.6 DISEÑO DEL CONTROLADOR DIFUSO

Se realiza el diseño de un controlador basado en la teoría previamente presentada acerca de lógica difusa. Se propone la implementación de una estructura PD para el presente controlador. Esto se lo hace con el objetivo de analizar posteriormente el rendimiento de este controlador difuso en comparación con el controlador cinemático convencional anteriormente diseñado. El diagrama de bloques que describe la estructura del sistema con el controlador difuso se presenta en la Figura 2.11.

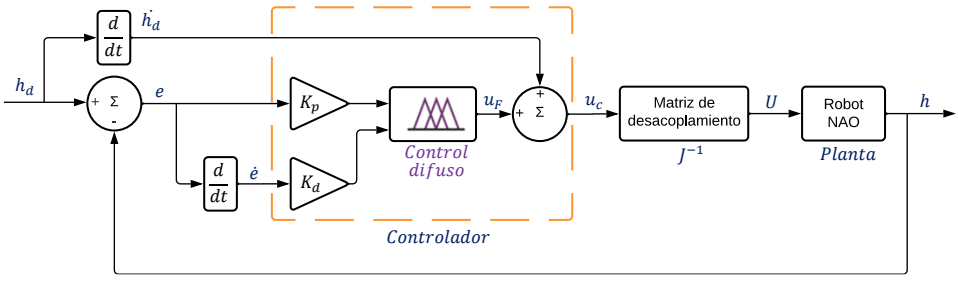


Figura 2.11. Diagrama de bloques del sistema de control del robot NAO usando un controlador PD difuso.

En primer lugar, se tiene en cuenta las entradas y salidas del controlador difuso. En este caso, como se observa en la Figura 2.11, cuenta con dos entradas (e y $\dot{e}$) y una salida (u_F). Posteriormente, se definen las funciones de membresía y variables lingüísticas de los

conjuntos difusos correspondientes para cada variable, así como el universo de discurso de cada una de éstas.

En el presente diseño se consideran 7 conjuntos difusos para cada variable (entradas y salida). Se proponen funciones de membresía de tipo triangular debido a que éstas presentan una menor carga computacional. El universo de discurso de cada variable se establece en el rango de -1 a 1 con el objetivo de obtener un controlador difuso normalizado unitariamente, y que sus entradas puedan ser escaladas por K_p y K_d, como se puede observar en la Figura 2.11.

A cada conjunto difuso, se le asignó su respectiva variable lingüística, cuyo nombre fue establecido tomando en cuenta la nomenclatura de la Tabla 2.1.

Tabla 2.1. Asignación de Variables Lingüísticas.

Variable Lingüística	**Significado**
NG	Negativo Grande.
NM	Negativo Mediano.
NP	Negativo Pequeño.
Z	Cero.
PP	Positivo Pequeño.
PM	Positivo Mediano.
PG	Positivo Grande.

Por ejemplo, la variable lingüística que identifica a los valores del universo del discurso muy cercanos a 1 es “PG”. Así mismo, la variable lingüística “Z” identifica a la función de membresía del conjunto difuso cuyos valores son muy cercanos a cero en el universo de discurso.

Dado que ambas entradas y la salida del controlador presentan el mismo universo de discurso y número de conjuntos difusos, se definen las mismas funciones de membresía triangulares para cada variable (e, $\dot{e}$ y u_F). En la Figura 2.12, se muestran las funciones de membresía definidas con sus respectivas variables lingüísticas. Es importante mencionar que “e”, “de y “uF”, corresponden al error, la derivada del error y la salida de control, respectivamente.

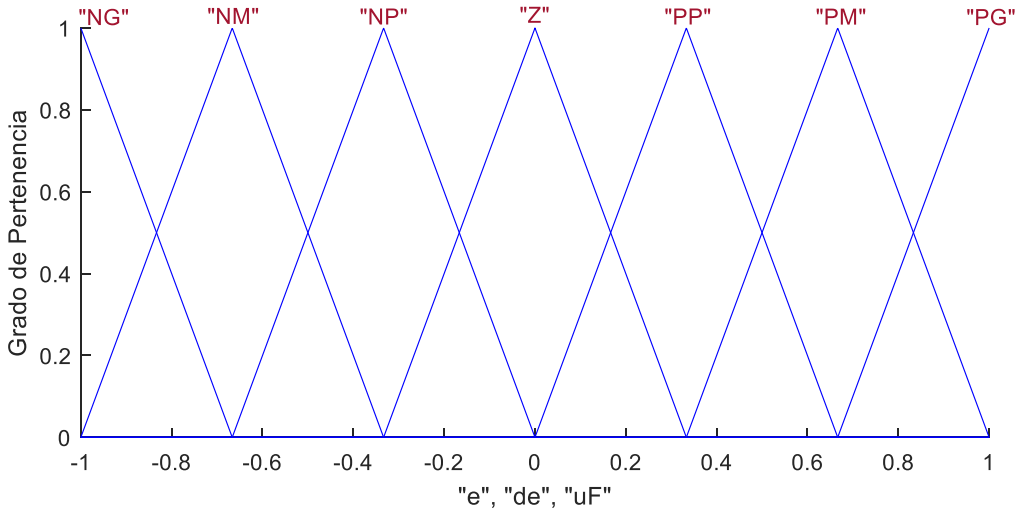


Figura 2.12. Funciones de membresía y variables lingüísticas de cada conjunto difuso.

Posteriormente, se establece la base de reglas del controlador difuso. Recordando que se cuenta con 7 funciones de membresía por cada entrada, se obtiene un total de 49 reglas difusas, siguiendo la ecuación (2.22).

$$número\ de\ reglas = (número\ de\ conjuntos\ difusos)^{número\ de\ entradas} \qquad (2.22)$$

En la Tabla 2.2, se presenta la base de reglas difusas tipo Mamdani con estructura SI – ENTONCES (*IF – THEN*).

Tabla 2.2. Base de reglas para el controlador difuso.

THEN Salida "uF"		***AND* Derivada de Error "de"**						
		NG	**NM**	**NP**	**Z**	**PP**	**PM**	**PG**
***IF* Error "e"**	**NG**	NG	NG	NG	NG	NM	NP	Z
	NM	NG	NG	NG	NM	NP	Z	PP
	NP	NG	NG	NM	NP	Z	PP	PM
	Z	NG	NM	NP	Z	PP	PM	PG
	PP	NM	NP	Z	PP	PM	PG	PG
	PM	NP	Z	PP	PM	PG	PG	PG
	PG	Z	PP	PM	PG	PG	PG	PG

Luego, se escoge un método de inferencia para la salida de control. En este caso, se selecciona el método de Mamdani, que corresponde a la operación difusa de intersección, es decir, el operador mínimo. Finalmente, se escoge el operador máximo como método de agregación, y el método del centroide para la etapa de desfuzzificación.

2.7 PROTOCOLO DE COMUNICACIÓN UDP

El Protocolo de Datagramas de Usuario, UDP (User Datagram Protocol), es un protocolo de comunicación sin conexión que forma parte de la familia de protocolos de internet (IP), por lo cual, su operación se da en la capa de transporte del modelo OSI. Fue definido por Postel en la RFC (Request For Comments) 768 en 1980 [46], [47].

UDP permite la trasmisión rápida de datagramas en redes IP. A diferencia de otros protocolos, como TCP, UDP no establece ni cierra una conexión entre el emisor y receptor. Esto implica que la velocidad de trasmisión de datos sea elevada debido a la ausencia de la sobrecarga asociada al establecimiento y cierre de conexiones. Sin embargo, y debido precisamente a la falta de aquel mecanismo de control de flujo de datos, el protocolo UDP tiende a presentar una alta probabilidad de que sus datos se pierdan o lleguen desordenadamente. Por tanto, la aplicación del protocolo UDP es adecuada siempre que se busque una comunicación rápida y sencilla de datos [47], [48].

En el presente proyecto, debido a las características de velocidad, se hace uso del protocolo UDP para establecer comunicación entre el controlador y el entorno de simulación del robot NAO.

2.8 IMPLEMENTACIÓN DEL SISTEMA DE CONTROL

A continuación, se describe el proceso de implementación del sistema de control y simulación del robot NAO, detallando las configuraciones necesarias que deben programarse en los diversos programas relevantes para el correcto funcionamiento del sistema.

2.8.1 CONFIGURACIÓN DEL CONTROLADOR EN SIMULINK

En Simulink, se lleva a cabo la implementación de los diagramas de bloques correspondientes a los controladores convencional y difuso, tal como se muestra en la Figura 2.6 y Figura 2.7, respectivamente. En la Figura 2.13, se muestra el diagrama de bloques implementado en Simulink con el controlador difuso y la comunicación establecida mediante protocolo UDP con el entorno de simulación del robot NAO.

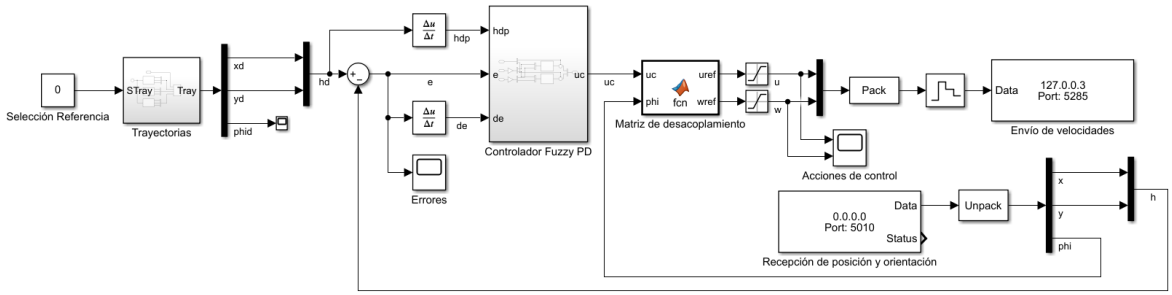


Figura 2.13. Diagrama de bloques con Controlador Difuso en Simulink.

El subsistema denominado “Trayectorias” de la Figura 2.13 consta de tres bloques de función de Matlab, los cuales han sido programados con las ecuaciones correspondientes a las tres trayectorias de referencia presentadas en la sección 2.4. La constante de selección de referencia permite elegir entre las tres trayectorias, siendo 0 para la trayectoria circular, 1 para la lemniscata y 2 para la cuadrada. En la Figura 2.14 se muestran los bloques que componen el subsistema de generación de trayectorias.

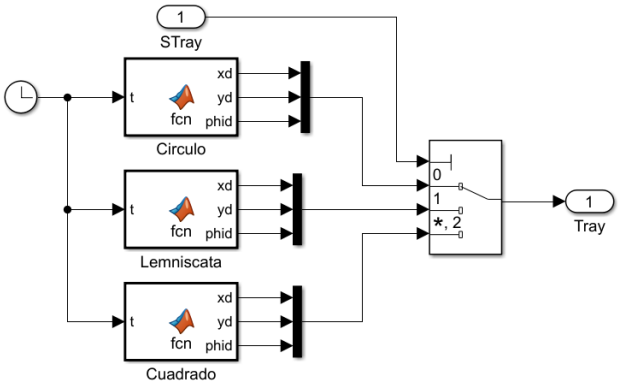


Figura 2.14. Subsistema de Generación de Trayectorias.

Posteriormente, siguiendo el diagrama de la Figura 2.13, se tiene el subsistema “Controlador Fuzzy PD”, mismo que contiene a los controladores difusos de cada componente de movimiento (ejes X y Y), como se observa en la Figura 2.15.

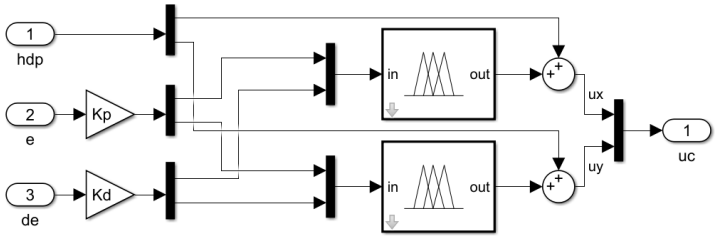


Figura 2.15. Subsistema de Controlador Difuso en Simulink.

Luego, la matriz de desacoplamiento es programada haciendo uso de una función de Matlab, tomando en cuenta la ecuación (2.18). Los saturadores a la salida de la matriz de desacoplamiento limitan las señales de control a las velocidades físicas máximas que el robot NAO es capaz de alcanzar.

Después, se tienen los bloques que establecen la comunicación UDP, como se muestran en la Figura 1.4. El bloque “Pack” permite el empaquetado en bytes de las señales de control, y el bloque “Unpack”, el desempaquetado de las señales de posición y orientación del robot. El envío de las señales de control se lo realiza a la tasa de refresco del robot NAO, por tanto, se hace uso del retenedor de orden cero para discretizar las velocidades a un tiempo de muestreo de 20 [ms]. Finalmente, los bloques “UDP Send” y “UDP Receive” son configurados con las direcciones IP y números de puerto para el envío y recepción de las señales, respectivamente, con el programa de Python.

2.8.1.1 Fuzzy Logic Toolbox

Los bloques de controlador difuso que se muestran en la Figura 2.15 se programan haciendo uso del “Fuzzy Logic Toolbox” de Matlab. En la Figura 2.16, se muestra la interfaz de configuración de mencionado Toolbox, llamada “Fuzzy Logic Designer”.

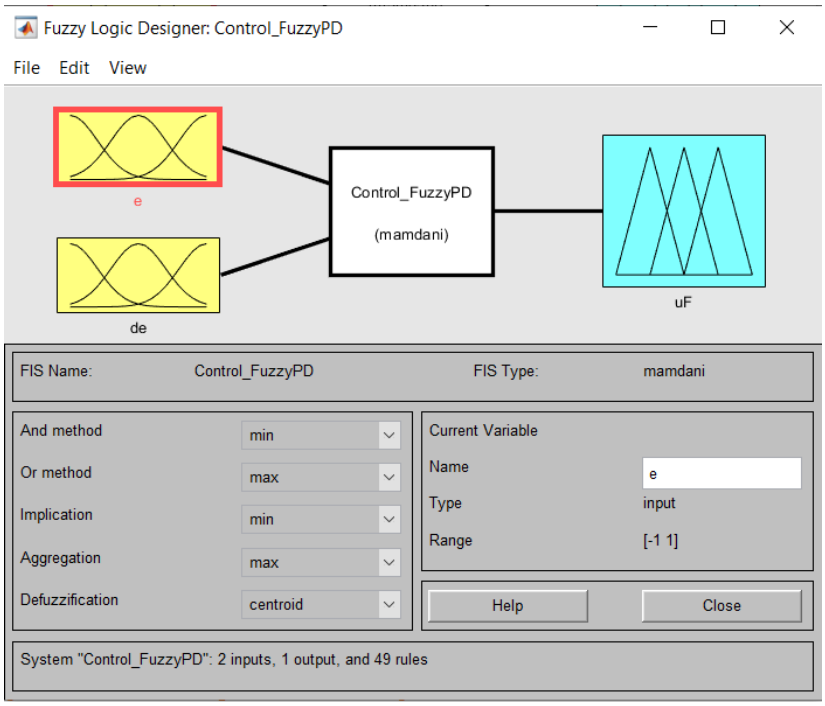


Figura 2.16. Fuzzy Logic Designer de Matlab.

Como se puede observar, éste permite la configuración de las múltiples características del controlador difuso: variables de entrada, salida, funciones de membresía, reglas difusas,

método de desfuzzificación, de inferencia y de agregado. Además, el "Fuzzy Logic Designer" permite observar de manera preliminar el comportamiento del controlador difuso ante las variaciones de sus entradas. En la sección ANEXOS, se describe de manera más detallada el uso del "Fuzzy Logic Toolbox".

2.8.2 PROGRAMA DE COMUNICACIÓN EN SPYDER

Como se había mencionado previamente, en la sección 1.4.7, Spyder es el entorno de desarrollo de Python que se escogió para programar los comandos necesarios que permitan la integración de la comunicación y enlazamiento entre Matlab/Simulink, el framework NAOqi y el entorno de simulación del robot NAO en CoppeliaSim. Es decir, el script de Python actuará como intermediario entre los distintos programas involucrados para el funcionamiento del presente proyecto. En la Figura 2.17, se observa, de manera simplificada, la arquitectura de comunicación y los comandos de programación necesarios en cada software para el funcionamiento de la comunicación en el proyecto.

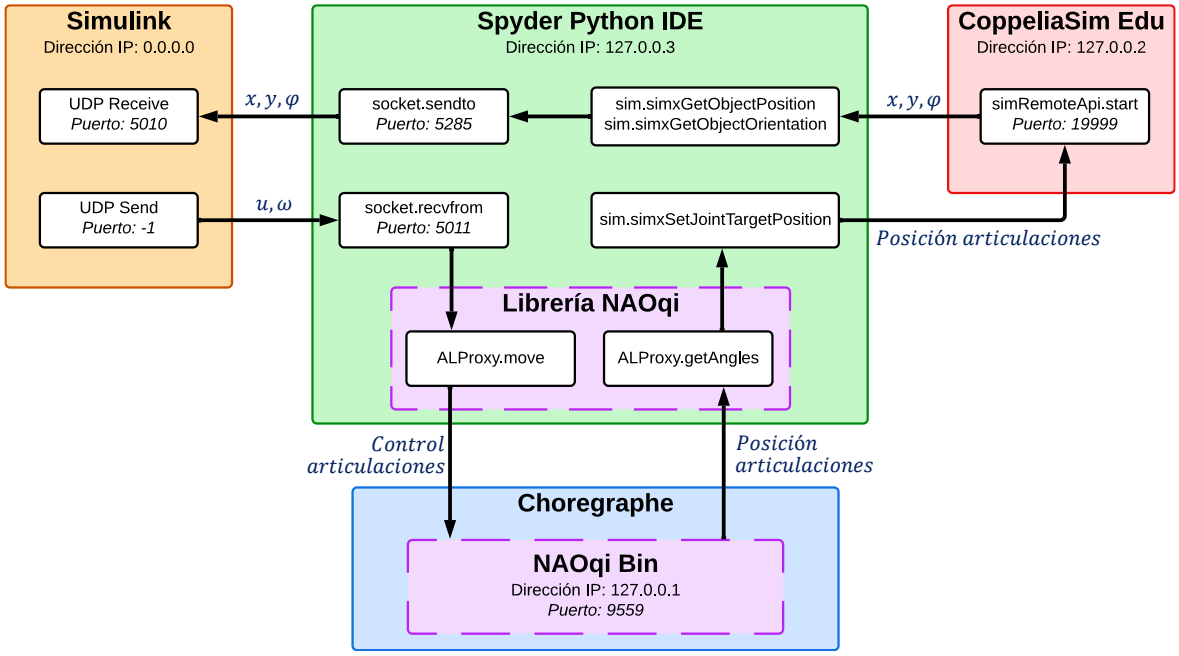


Figura 2.17. Arquitectura de comunicación entre programas.

Como se aprecia en la Figura 2.17, Simulink envía los datos de velocidades de control mediante el bloque *UDP Send* hacia el programa de Python, donde recibe aquella información haciendo uso de la función *rcvfrom* de la librería *socket* para comunicaciones. Posteriormente, aquellas señales de control son enviadas hacia el robot virtual que se encuentra en NAOqi Bin – Choregraphe para ejecutar su movimiento. Este envío se lo realiza mediante la función *move* del módulo *ALMove,* perteneciente al objeto *ALProxy de* la librería *NAOqi* en Python. Mediante la ejecución del comando *getAngles* en Python, se

recibe la posición angular de cada una de las articulaciones del robot virtual. Después, para enviar aquellos datos de posición angular de las articulaciones hacia el entorno de simulación del robot NAO en CoppeliaSim, se hace uso de la función *simxSetTargetPosition* por cada articulación del robot. Esta última función de Python mencionada se encuentra en la librería de comunicación con CoppeliaSim, denominada *sim*. Para que se establezca comunicación con CoppeliaSim, se emplea el comando *simRemoteApi.start,* en el script de un elemento fijo del entorno de simulación.

De esta manera, el robot NAO de CoppeliaSim ejecutará su movimiento en el entorno de simulación. Luego, para proporcionar retroalimentación al sistema de control sobre la posición y orientación del robot, se emplean las funciones *simxGetObjectPosition* y *simxGetObjectOrientation* en Python. Después, estos datos de posición y orientación del robot son enviados hacia Simulink utilizando la función *sendto* de Python. Finalmente, en Simulink, se recibe los datos mediante el bloque de función ya mencionado en la anterior sección, *UDP Receive*.

Cada programa empleado cuenta con una dirección IP de la misma red para establecer comunicación entre ellos. Del mismo modo, cada función de envío y recepción de datos tiene asignado un puerto de comunicación para su funcionamiento. La especificación de dirección IP 0.0.0.0 y puerto -1 en los bloques UDP de Simulink significa que la comunicación se realizará en alguna red dentro de la misma máquina (computador). Por tanto, se asignarán de manera automática una dirección IP y un puerto disponible.

Un objeto *ALProxy* presenta un comportamiento que depende del módulo con el que se lo define. Cada módulo contiene diversas funciones o métodos que le permiten al robot NAO ejecutar acciones y, a su vez, obtener información. NAOqi contiene un gran número de módulos con los que se pueden programar múltiples actividades para que el robot NAO las ejecute. En este proyecto se ha empleado exclusivamente el módulo denominado *ALMotion*. Este módulo posibilita la ejecución de funciones, métodos o comandos que permiten al robot NAO llevar a cabo movimientos y enviar retroalimentación sobre los mismos hacia Python. Es importante tener en cuenta que el módulo *ALMotion* se ejecuta a una frecuencia de 50 [Hz], es decir, en ciclos de 20 [ms] [49], [50].

Para la ejecución de la caminata y recepción de posiciones angulares del robot NAO virtual se hace uso específicamente de las funciones *ALMotionProxy.move* y *ALMotionProxy.getAngles*. La función *ALMotionProxy.move*, con las velocidades lineal y angular como argumentos, dirige el movimiento de caminata del robot NAO según estas

señales de control, semejante a un robot móvil [41]. Por otro lado, la función *ALMotionProxy.getAngles* se encarga de adquirir las posiciones angulares de cada articulación del robot NAO virtual [51].

Una vez revisados aquellos conceptos importantes, en la Figura 2.18, se observa el diagrama de flujo general del programa de Python implementado en Spyder.

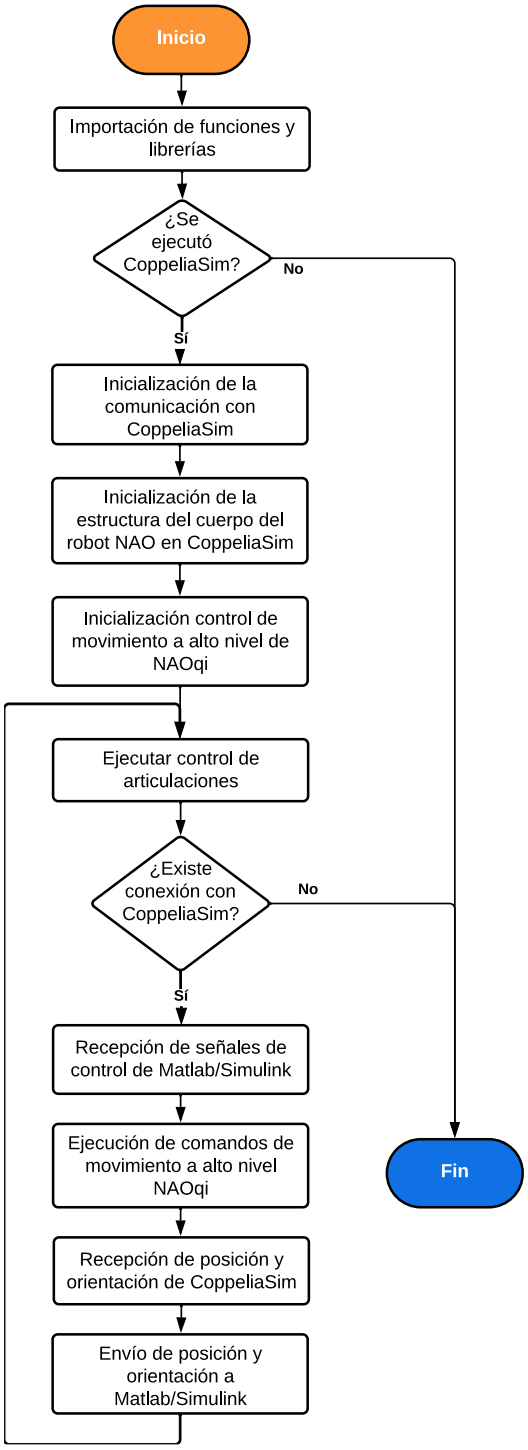


Figura 2.18. Diagrama de flujo del programa de Python en Spyder.

2.8.3 DESARROLLO DEL ENTORNO DE SIMULACIÓN EN COPPELIASIM EDU

CoppeliaSim Edu es un software que permite el desarrollo de entornos de simulación robóticos, como el presentado en la Figura 1.5, donde también se aprecia la interfaz de CoppeliaSim Edu.

Para la creación de un nuevo escenario, se debe acceder en la cinta de opciones a “File” y dar clic en “New Escene”. Una vez creada la escena, se pueden agregar elementos al entorno, como muebles, herramientas, vehículos y robots. Para agregar un robot al escenario, se accede a la carpeta “robots” del “Model Browser”. En la sección de robots móviles, se encuentra el robot humanoide NAO, mismo que se empleará en el proyecto actual. Para agregar el robot al entorno de trabajo, simplemente se arrastra el modelo del robot y se lo suelta en el área designada.

De manera predeterminada, el robot NAO está equipado con un script de CoppeliaSim que le permite realizar ciclos continuos de movimientos, que consisten en desplazamientos en línea recta y giros aproximados de 90°. Por lo tanto, es necesario deshabilitar ese script seleccionando la opción "Scripts" en la barra de herramientas vertical, la cual se encuentra en la parte izquierda de la interfaz de la Figura 1.5.

Por último, para establecer la comunicación con el programa de Python, se utiliza un elemento adicional en el espacio de trabajo. Se coloca un cuboide y se agrega el comando *simRemoteApi.start* en su script de CoppeliaSim correspondiente. En el argumento de dicha función, se especifica el puerto para la comunicación. Este comando permite la inicialización de la comunicación entre CoppeliaSim y una API remota, que en este caso es el programa de Python en Spyder.

En el presente proyecto, se programa la posición inicial del robot NAO en el origen del sistema de coordenadas del entorno de simulación de CoppeliaSim Edu, como se observa en la Figura 2.19.

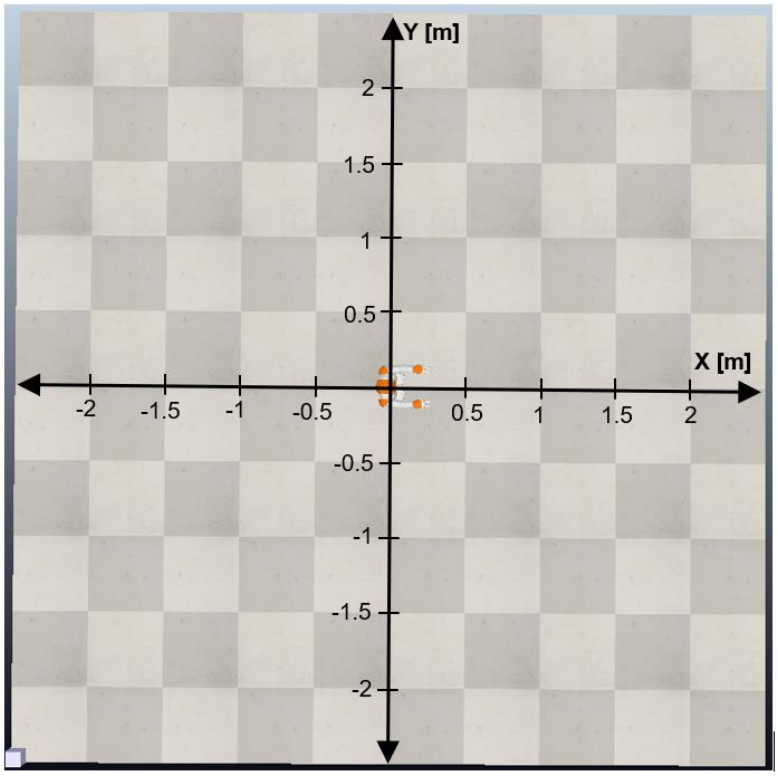


Figura 2.19. Sistema de coordenadas del entorno de simulación en CoppeliaSim Edu.

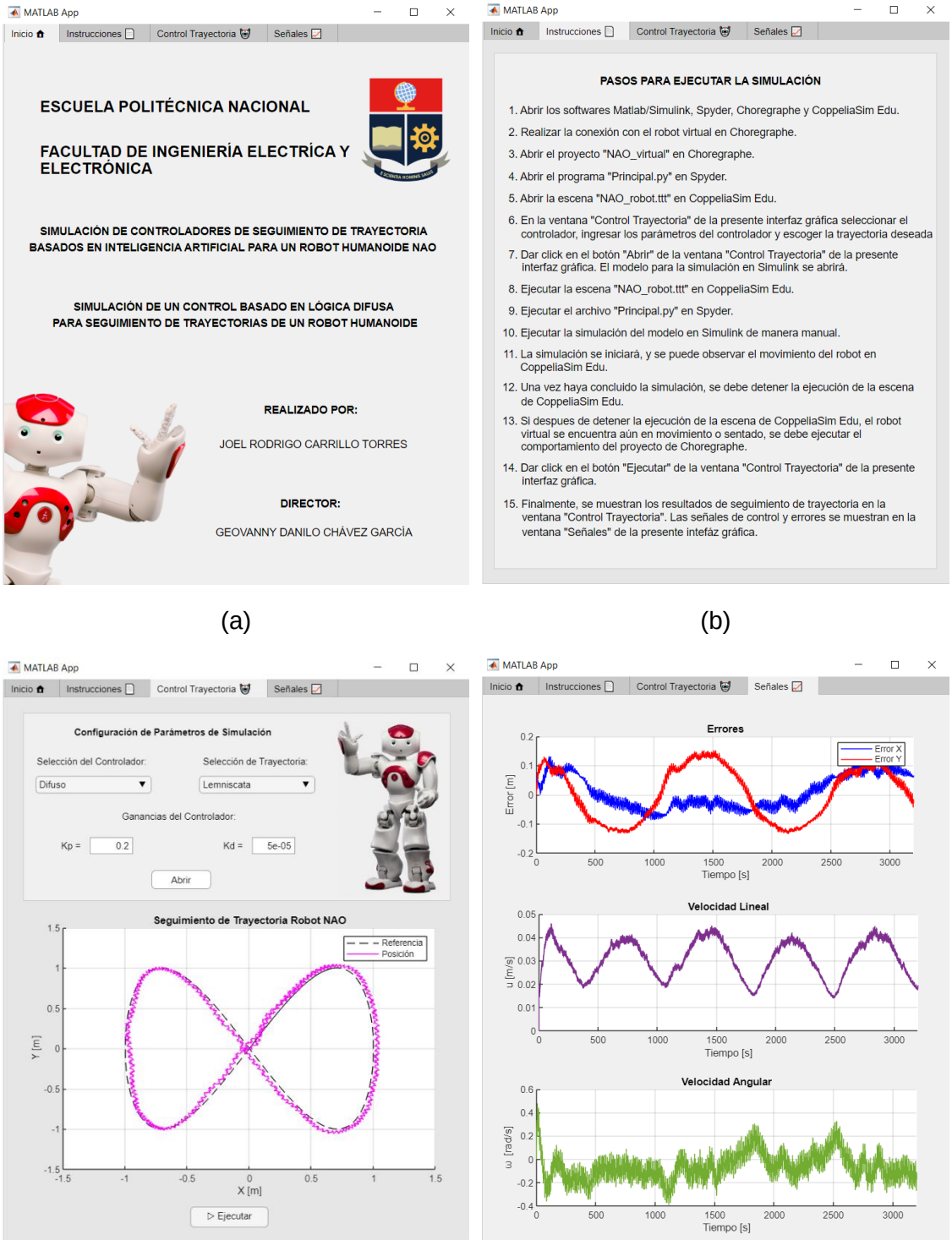
2.9 DISEÑO DE INTERFAZ GRÁFICA

Se realiza el diseño de una interfaz gráfica que permita al usuario interactuar de una manera más simple y amigable con la simulación del control de seguimiento de trayectoria del robot humanoide NAO. Esta interfaz posibilita la configuración de los parámetros del controlador, la elección de la trayectoria deseada, y la observación rápida y completa de todos los resultados provenientes de la simulación del sistema de control propuesto.

Para el diseño de la interfaz, se considera la estandarización ISO 9241, titulada como *Ergonomía de la interacción humano – sistema*. De manera específica, la norma ISO 9241 – 161 establece determinados requisitos, consideraciones y recomendaciones respecto al aspecto visual que debe poseer una interfaz de usuario. En base a las recomendaciones clave que contiene mencionado estándar, se puede concluir que una interfaz de usuario debe ser simple, intuitiva, no contener elementos en exceso, presentar colores que no provoquen un agotamiento visual y evitar el uso de distintos tipos de fuentes y tamaños de letra [52].

La interfaz desarrollada se encuentra dividida en 4 ventanas, a las cuales se puede acceder mediante pestañas ubicadas en la parte superior de la interfaz. En la Figura 2.20 se observa

la interfaz de usuario desarrollada para el presente proyecto, compuesta de cuatro ventanas.



(a) (b)

(c) (d)

Figura 2.20. Interfaz Gráfica desarrollada. (a) Ventana de Inicio. (b) Ventana de Instrucciones. (c) Ventana de Control de Trayectoria. (d) Ventana de Señales.

La primera ventana de la interfaz de usuario diseñada corresponde al Inicio, donde se detalla el nombre de la universidad, la facultad de la carrera, el título del proyecto, la componente específica desarrollada y los nombres del autor y director del proyecto. En la segunda ventana, se muestran las instrucciones que se deben seguir con el objetivo de ejecutar de manera satisfactoria la simulación del control de seguimiento de trayectorias del robot NAO.

Luego, en la tercera ventana, denominada “Control Trayectoria”, se tiene un bloque que permite configurar los parámetros del controlador que se desea simular. En esta ventana, se puede escoger entre el controlador difuso propuesto y el controlador PD convencional. Igualmente, se puede modificar las ganancias de cada controlador y seleccionar la trayectoria que se desea simular. Dentro de esta misma ventana, se encuentra un gráfico que mostrará la respuesta del seguimiento de trayectorias, una vez que la simulación haya finalizado. Las funciones de los botones que se encuentran en esta ventana se especifican en la sección ANEXOS.

Finalmente, la cuarta ventana desarrollada se llama “Señales” y contiene las gráficas de las señales más importantes que se obtienen del sistema de control propuesto, como las señales de control (velocidad lineal y velocidad angular) y errores de posición (en el eje X y en el eje Y).

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En la presente sección se presentan y analizan los resultados obtenidos de la simulación del control basado en lógica difusa aplicado al seguimiento de trayectorias del robot humanoide NAO. Se muestra gráficamente el comportamiento del sistema de control dadas distintas trayectorias de referencia. Posteriormente, se compara el desempeño del controlador difuso diseñado con el desempeño de un controlador PD convencional, basándose en el análisis de varios índices que rendimiento. Partiendo de los resultados presentados, se emiten las respectivas conclusiones y recomendaciones con respecto a la realización del presente proyecto.

3.1 RESULTADOS

Tanto los resultados del controlador difuso como los del controlador PD convencional se derivan de los parámetros de ganancia que se presentan en la Tabla 3.1.

Tabla 3.1. Parámetros del controlador.

Parámetro de ganancia	**Trayectoria**		
	Circular	***Lemniscata***	***Cuadrada***
K_p	0.2	0.2	0.16
K_d	0.00005	0.00005	0.00005

Se exponen las respuestas de ambos sistemas de control y se procede a contrastar su comportamiento a través de gráficas que ilustran las diversas señales generadas por cada sistema. Estas gráficas muestran específicamente la posición del robot en el plano XY, su orientación, los errores de posición y las señales de control (velocidades lineal y angular). De igual manera, se presentan diagramas de barras indicando la magnitud de los índices de desempeño ISE, ISU y TVu de cada controlador.

3.1.1 RESULTADOS TRAYECTORIA CIRCULAR

En la Figura 3.1 se presenta la respuesta de la simulación del robot NAO ante una referencia de trayectoria circular de 1 [m] de radio. En estas gráficas se puede observar que ambos controladores cumplen con el seguimiento de la circunferencia deseada. Sin embargo, se puede notar que, en la orientación del robot, el controlador difuso presenta un mejor comportamiento, con una menor cantidad de oscilaciones en comparación con el

controlador PD. Además, el controlador difuso presenta un menor rango de errores ante señales de control más elevadas que el controlador convencional. Con respecto a la velocidad angular, se puede apreciar que, en el caso difuso, esta señal presenta mayor cantidad de oscilaciones.

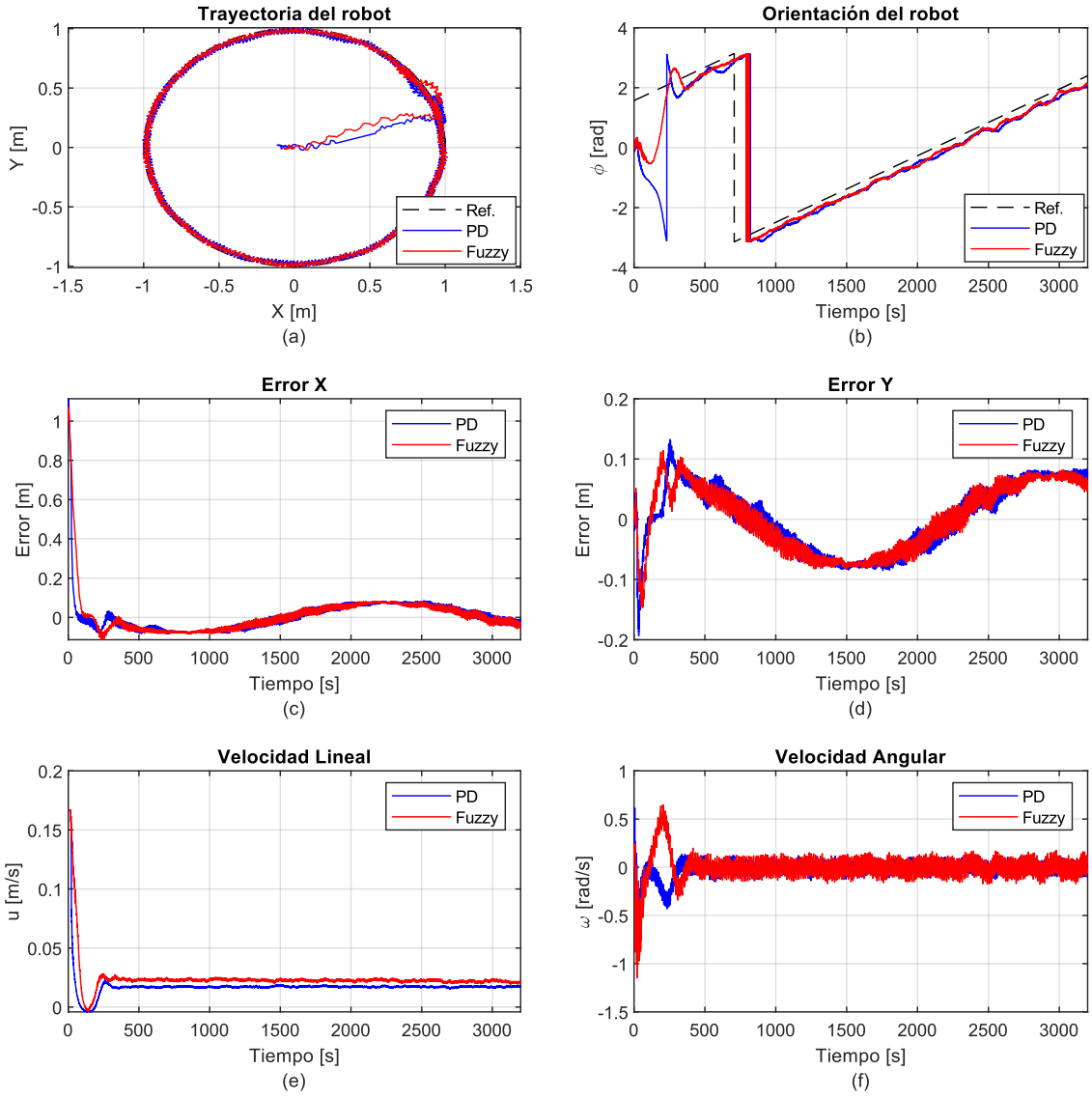


Figura 3.1. Respuesta del robot humanoide NAO con la referencia de trayectoria circular. (a) Posición en el plano XY. (b) Orientación. (c) Error de posición en eje X. (d) Error de posición en el eje Y. (e) Velocidad lineal. (f) Velocidad angular.

Posteriormente, en la Figura 3.2 se muestra la comparación entre los índices de desempeño del controlador difuso y el controlador convencional ante los resultados de la Figura 3.1. Se puede apreciar que el valor de ISE en el eje Y del controlador difuso es menor que el valor obtenido del controlador convencional, lo que significa que el error de posición en el eje Y disminuye más con el controlador difuso. Del mismo modo, el valor de

TVu de la velocidad angular es menor en el caso del controlador difuso, lo que significa que dicha acción de control es más suave que en el controlador convencional.

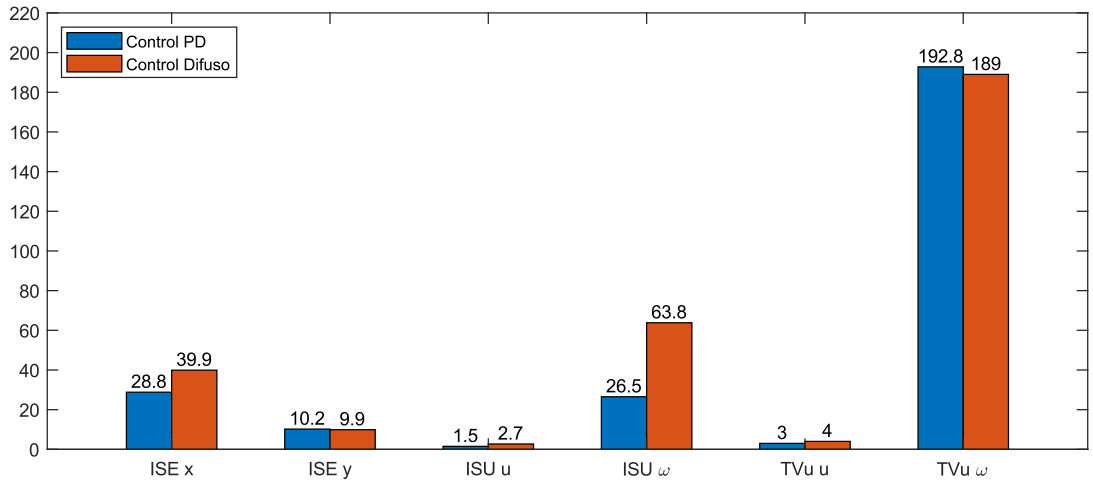


Figura 3.2. Índices de desempeño del controlador ante una referencia de trayectoria circular.

3.1.2 RESULTADOS TRAYECTORIA LEMNISCATA

En la Figura 3.3 se aprecian los indicadores de desempeño ante los resultados de la trayectoria Lemniscata de la Figura 3.4. Se puede observar que la respuesta del controlador difuso presenta un ISE de posición en X y un TVU de velocidad angular menores. Sin embargo, los demás índices de desempeño demuestran que, para esta trayectoria, la implementación de un controlador PD tradicional es más conveniente para obtener una mayor eficiencia.

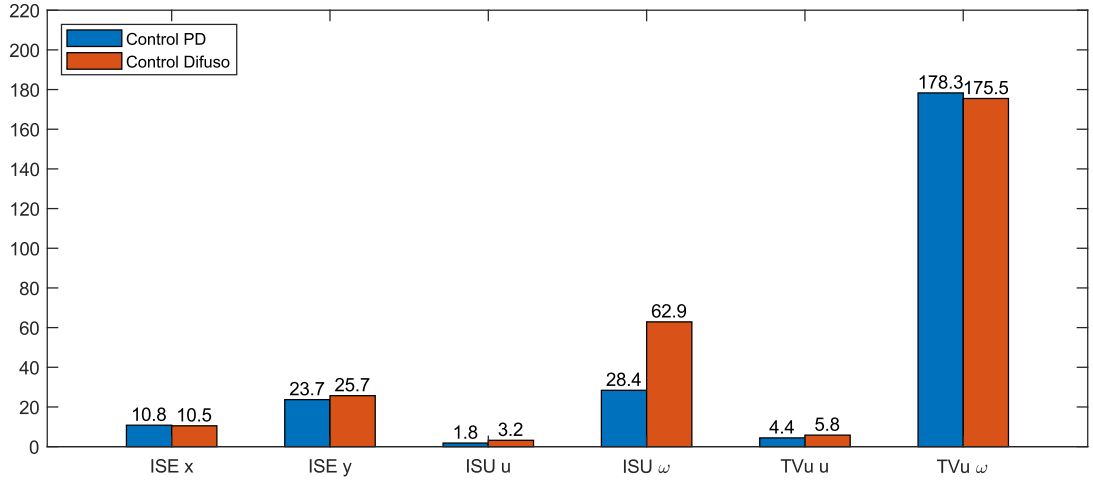


Figura 3.3. Índices de desempeño del controlador ante una referencia de trayectoria lemniscata.

En la Figura 3.4 se presenta la respuesta de la simulación del robot NAO ante una referencia de trayectoria lemniscata de 1 [m] de radio. Se puede notar que, en la posición

del robot, el controlador difuso presenta una mayor cantidad de oscilaciones y un mayor rango de variación de los errores en comparación con el controlador PD. De igual manera, en la señal de control de velocidad angular, se puede apreciar que el controlador difuso presenta mayor cantidad de oscilaciones que el controlador convencional. Sin embargo, el controlador difuso presenta velocidades lineales más elevadas sin comprometer un aumento importante en el rango de errores de posición.

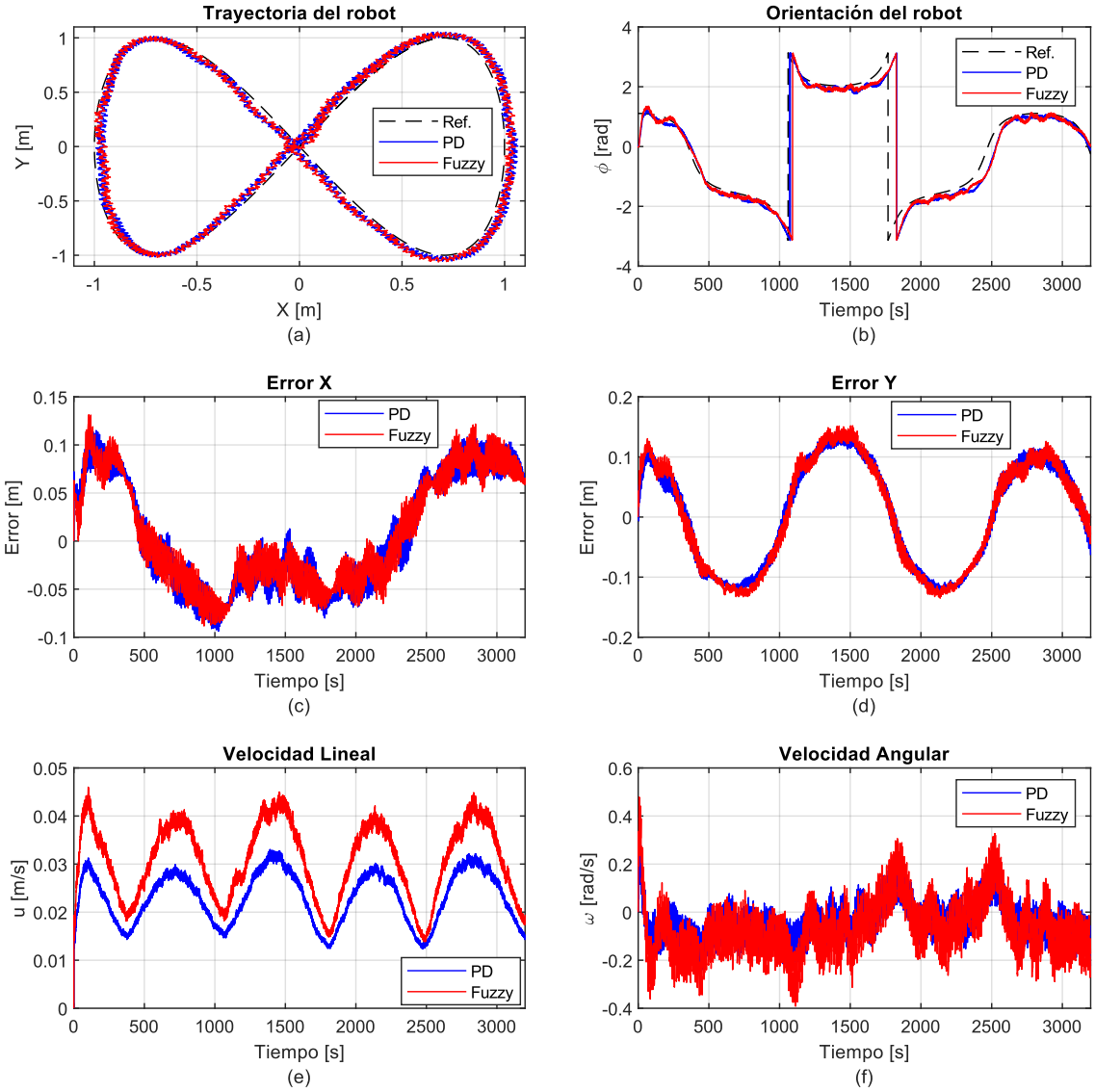


Figura 3.4. Respuesta del robot humanoide NAO con la referencia de trayectoria lemniscata. (a) Posición en el plano XY. (b) Orientación. (c) Error de posición en eje X. (d) Error de posición en el eje Y. (e) Velocidad lineal. (f) Velocidad angular.

3.1.3 RESULTADOS TRAYECTORIA CUADRADA

En la Figura 3.5 se presenta la respuesta de la simulación del robot NAO ante una referencia de trayectoria cuadrada de 2 [m] de lado. Se puede notar que ambos

controladores presentan respuestas muy similares. Sin embargo, al igual que en las anteriores trayectorias, la velocidad lineal y angular del controlador difuso se encuentra en valores más elevados que el controlador PD. Esto provoca que los índices de desempeño ISU y TVu de ambas señales sean mayores en el controlador difuso, como se observa en la Figura 3.6. En la orientación del robot, se puede notar que la respuesta del controlador difuso presenta sobrepicos mayores en comparación con la respuesta del controlador convencional. Lo mismo ocurre en los errores de movimiento tanto del eje X como del eje Y, la respuesta del controlador difuso presenta un sobreimpulso mayor al momento de que el robot llega a una esquina de la trayectoria cuadrada. Esto provoca que el robot describa una curva más pronunciada en cada esquina del cuadrado de referencia.

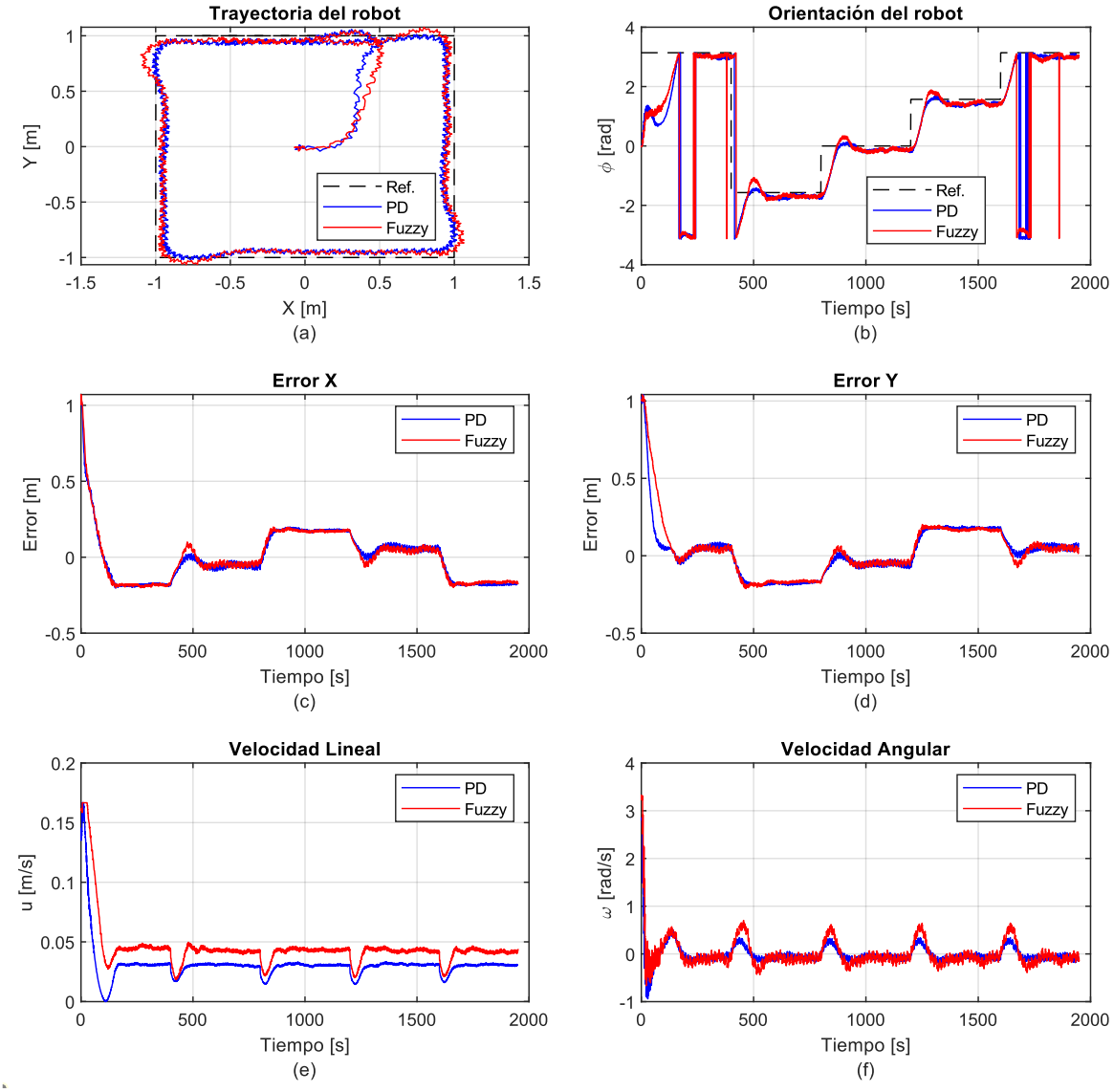


Figura 3.5. Respuesta del robot humanoide NAO con la referencia de trayectoria cuadrada. (a) Posición en el plano XY. (b) Orientación. (c) Error de posición en eje X. (d) Error de posición en el eje Y. (e) Velocidad lineal. (f) Velocidad angular.

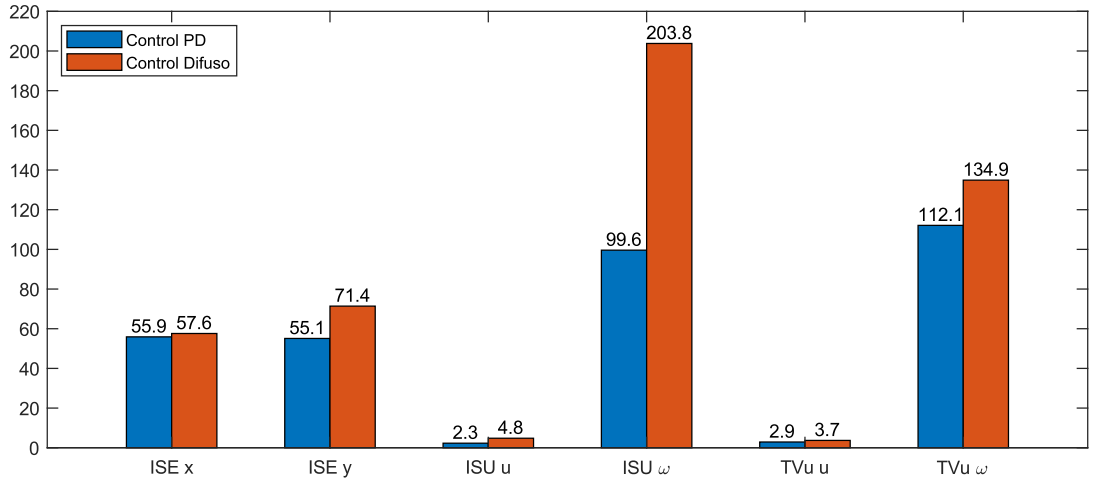


Figura 3.6. Índices de desempeño del controlador ante una referencia de trayectoria cuadrada.

Como se puede observar en los resultados obtenidos, los índices de desempeño del controlador difuso son generalmente mayores en comparación con los índices del controlador convencional. Esto significa que, en el controlador difuso, el esfuerzo de control es mayor, las señales de control son más agresivas y los errores de posición son más grandes. Lo que advierte que, pese a cumplir con el seguimiento de trayectorias, la aplicación de este controlador difuso no es la más óptima en el sistema de control propuesto para el robot NAO.

3.2 CONCLUSIONES

- Se realizó un estudio bibliográfico acerca de las características principales del robot humanoide NAO con el objetivo de comprender su funcionamiento, capacidades estructurales y software del robot. Partiendo de este estudio, se inició el diseño de un sistema de control que se adapte a las especificaciones del robot y funcione apropiadamente. Del mismo modo, se estudió los fundamentos teóricos de inteligencia artificial y lógica difusa con el objetivo de diseñar un controlador difuso y enfocarlo a la aplicación propuesta en el presente proyecto. Finalmente, se revisó los conceptos principales sobre el funcionamiento de la comunicación UDP y los distintos programas que permiten integrar el sistema de control propuesto para el robot NAO.

- Se diseñó e implementó un controlador basado en lógica difusa para el seguimiento de trayectorias de un robot humanoide NAO partiendo del modelo cinemático de un robot móvil, mismo que se adapta acertadamente al comportamiento del robot humanoide mencionado cuando se aplica un control a alto nivel. El controlador

diseñado mantiene una estructura PD con el propósito de ajustar sus parámetros para influir en las características del control, tales como minimizar el error en estado estable y reducir al máximo el sobreimpulso en las respuestas del controlador. Sin embargo, se puede notar que, debido a la naturaleza oscilatoria del sistema, la componente derivativa del controlador debe ser muy baja para un correcto seguimiento de trayectorias. Entre las ventajas principales de la implementación de un controlador difuso se tiene que éste no necesita de un modelo de la planta para ser diseñado, y que cuenta con la capacidad de disminuir los efectos de las no linealidades del sistema.

- El diseño de un controlador difuso se lo realiza de manera secuencial y sistemática, es decir, es un proceso que cuenta con múltiples etapas en las cuales se va estructurando y definiendo el comportamiento que éste mantendrá. Por tanto, la eficiencia y precisión del controlador son relativas a la manera en la que el diseñador lo configure. Por ejemplo, la cantidad de conjuntos difusos, el tipo de funciones de membresía, el universo del discurso, el método de inferencia, el método de desfuzzificación, entre otros, pueden variar dependiendo de la voluntad y experiencia del diseñador. Esto hace del diseño y sintonización del control difuso un verdadero arte.

- Se evalúo el desempeño del controlador basado en lógica difusa desarrollado para el seguimiento de trayectorias del robot NAO, y se lo comparó con el de un controlador convencional tipo PD con el propósito de analizar la eficiencia de implementación de mencionado control difuso en la aplicación presentada en el presente proyecto. En base a los resultados obtenidos, se puede concluir que el comportamiento de ambos controladores es muy similar. Sin embargo, con los índices de desempeño calculados, cualitativamente se verifica que el controlador convencional tipo PD presenta una mejor eficiencia en comparación con el controlador difuso, arrojando errores menores y acciones de control más suaves. Esto quiere decir que, en la presente aplicación, el controlador difuso no es precisamente la mejor opción para implementar en el sistema de control, dado que simplemente con un controlador PD se obtienen mejores resultados.

3.3 RECOMENDACIONES

- Revisar la información presentada en los ANEXOS del presente documento. En esta sección se detalla información acerca de los requerimientos necesarios para un adecuado funcionamiento del sistema. Entre los requisitos más importantes se tiene que la computadora desde la cual se realizará la simulación y control del sistema cuente con tarjeta gráfica y al menos 16 [GB] de memoria RAM. Del mismo modo, la versión instalada del software Matlab debe ser la R2022a o cualquier otra anterior. Esto se debe a que únicamente estas versiones tienen compatibilidad con Python 2.7.

- Es importante realizar una revisión bibliográfica acerca de las características importantes tanto de hardware como de software del robot NAO. De esta forma, se puede diseñar un sistema de control que se adapte precisamente a las características del robot. Por ejemplo, se toman en cuenta las velocidades de funcionamiento máximas del robot para incluir en el diseño a los saturadores, los cuales evitan que las señales de control alcancen valores ante los cuales el robot llegue a un estado de operación máxima, estado en el cual sus actuadores pueden resultar afectados. Igualmente, se considera una tasa de envío de datos de 20 [ms] debido a la operación del módulo de movimiento del software del robot.

- Para futuros proyectos de investigación, se puede diseñar un controlador difuso con funciones de membresía diferentes entre sí, distintos métodos de desfuzzificación o de inferencia, de manera que se puedan comparar con los resultados del presente proyecto. Igualmente, se pueden integrar más componentes de los que existen en el robot NAO, como las cámaras de visión y los sensores infrarrojos. Esto permitirá implementar nuevos algoritmos de control que incluyan, además del seguimiento de trayectorias, un sistema de evasión de obstáculos o reconocimiento de objetivos.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] A. M. Porcelli, “La inteligencia artificial y la robótica: sus dilemas sociales, éticos y jurídicos,” *Derecho Glob. Estud. Sobre Derecho Justicia*, vol. 6, no. 16, pp. 49–105, 2020, doi: 10.32870/dgedj.v6i16.286.

[2] E. Burns and G. Lawton, “Definition Artificial Intelligence (AI),” *Enterprise AI*, 2023. https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence (accessed May 30, 2023).

[3] L. D. Urquhart, D. Reedman-Flint, and N. Leesakul, “Responsible Domestic Robotics: Exploring Ethical Implications of Robots in the Home,” *IRPN Sci.*, 2018, doi: 10.1108/JICES-12-2018-0096.

[4] G. Singh and V. K. Banga, “Robots and its types for industrial applications,” *Mater. Today Proc.*, vol. 60, pp. 1779–1786, Jan. 2022, doi: 10.1016/j.matpr.2021.12.426.

[5] J. Ruiz de Garibay Pascual, “Robótica: Estado del arte,” *Univ. Deuston*, vol. 54, p. 61, 2006.

[6] F. Rubio, F. Valero, and C. Llopis-Albert, “A review of mobile robots: Concepts, methods, theoretical framework, and applications,” *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, p. 1729881419839596, Mar. 2019, doi: 10.1177/1729881419839596.

[7] A. Sheth, S. Bhosale, and M. Burondkar, “Research Paper on Robotics-New Era,” *Contemp. Res. India*, pp. 257–261, Apr. 2021.

[8] A. O. Baturone, *Robótica: Manipuladores y Robots Móviles*. Barcelona, Spain: Marcombo, 2005. [Online]. Available: https://books.google.com.ec/books?hl=en&lr=&id=TtMfuy6FNCcC&oi=fnd&pg=PR15&dq=Rob%C3%B3tica&ots=33OYG2u3bP&sig=CyYRR-jo4OUTG_Ydwe3u_Odi6sw&redir_esc=y#v=onepage&q&f=false

[9] M. A. Miskam, S. Shamsuddin, M. R. A. Samat, H. Yussof, H. A. Ainudin, and A. R. Omar, “Humanoid robot NAO as a teaching tool of emotion recognition for children with autism using the Android app,” in *2014 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, Nov. 2014, pp. 1–5. doi: 10.1109/MHS.2014.7006084.

[10] J. M. Mendel, “Fuzzy logic systems for engineering: a tutorial,” *Proc. IEEE*, vol. 83, no. 3, pp. 345–377, Mar. 1995, doi: 10.1109/5.364485.

[11] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, Tercera. New Jersey, United States of America: Pearson Education, Inc., 2010.

[12] B. Griffen, “Robots bring flexibilty, speed to confectionery industry,” *Candy Industry*, Jan. 24, 2019. https://www.snackandbakery.com/articles/105111-

robots-bring-flexibilty-speed-to-confectionery-industry (accessed Aug. 19, 2023).

[13] H. Monaghan, “Mars Exploration Rover (Opportunity),” *NASA*, Jul. 20, 2014. http://www.nasa.gov/directorates/heo/scan/services/missions/solarsystem/MEROppy.html (accessed Aug. 19, 2023).

[14] Y. Amer, S. singh, and L. T. T. Doan, “Partial soft body robots - a literature review,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 715, no. 1, p. 012092, Jan. 2020, doi: 10.1088/1757-899X/715/1/012092.

[15] “Nao - ROBOTS: Your Guide to the World of Robotics.” https://robots.ieee.org/robots/nao/ (accessed Mar. 14, 2023).

[16] Aldebaran, “NAO - Documentation — Aldebaran 2.8.7.4 documentation,” *Softbank Robotics Documentation*. http://doc.aldebaran.com/2-8/home_nao.html (accessed Aug. 10, 2023).

[17] Aldebaran, “Locomotion control — Aldebaran 2.5.11.14a documentation,” *Softbank Robotics Documentation*. http://doc.aldebaran.com/2-5/naoqi/motion/control-walk.html#control-walk (accessed Sep. 14, 2023).

[18] D. Gouaillier *et al.*, “Mechatronic design of NAO humanoid,” in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 769–774. doi: 10.1109/ROBOT.2009.5152516.

[19] 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.

[20] M. K. Ireland, M. S. Orosz, J. G. Brisson, A. Desideri, and S. Quoilin, “Dynamic Modeling and Control System Definition for a Micro-CSP Plant Coupled With Thermal Storage Unit,” in *Volume 3B: Oil and Gas Applications; Organic Rankine Cycle Power Systems; Supercritical CO2 Power Cycles; Wind Energy*, Düsseldorf, Germany: American Society of Mechanical Engineers, Jun. 2014, p. V03BT26A016. doi: 10.1115/GT2014-27132.

[21] J. Fernández de Cañete, C. Galindo, and I. G. Moral, “Introduction to Control Systems,” in *System Engineering and Automation: An Interactive Educational Approach*, Berlin, Heidelberg: Springer, 2011, pp. 137–165. doi: 10.1007/978-3-642-20230-8_5.

[22] O. Camacho, A. Rosales, and F. Rivas, *Control de Procesos*, Primera Edición. Quito, Ecuador: EPN Editorial, 2020.

[23] “Introduction to Robotic Control Systems,” *ROBOTICS TECHNICIAN TRAINING Online Education Program*, Oct. 14, 2021. https://www.onlinerobotics.com/news-blog/introduction-robotic-control-systems (accessed Jun. 21, 2023).

[24] J. R. Ortiz del Castillo, “Robustez en Controladores Realimentados,” Instituto Tecnológico de Celaya, Celaya, México, 2005. [Online]. Available: http://www.iqcelaya.itc.mx/~richart/TesisDoctorado/2005%20Ort%C3%ADz%20del%20Castillo.pdf

[25] M. Shahbazzadeh, S. J. Sadati, and S. Minagar, “Trajectory tracking control for mobile robots considering position of mass center,” *Optim. Control Appl. Methods*, vol. 42, no. 6, pp. 1542–1555, 2021, doi: 10.1002/oca.2744.

[26] J. Pérez Porto and M. Merino, “Trayectoria - Qué es, tipos, definición y concepto.,” *Definición.de*, Mar. 31, 2010. https://definicion.de/trayectoria/ (accessed Jun. 11, 2023).

[27] E. Mariscal-García, “Planeación y Seguimiento de Trayectorias de Robots Móviles en una Simulación de un Ambiente Real,” *Ra Ximhai*, pp. 177–200, Apr. 2005, doi: 10.35197/rx.01.01.2005.12.EM.

[28] R. C. Dorf and R. H. Bishop, *Sistemas de control moderno*. Pearson Educación, 2005.

[29] A. Rodríguez-Mariano, G. Reynoso-Meza, D. E. Páramo-Calderón, E. Chávez-Conde, M. A. García-Alvarado, and J. Carrillo-Ahumada, “Análisis del desempeño de controladores lineales sintonizados en diferentes estados estacionarios del biorreactor de Cholette mediante técnicas de decisión multi-criterio,” *Rev. Mex. Ing. Quím.*, vol. 14, no. 1, pp. 167–204, Apr. 2015.

[30] A. Alkhayyat *et al.*, “Fuzzy logic, genetic algorithms, and artificial neural networks applied to cognitive radio networks: A review,” *Int. J. Distrib. Sens. Netw.*, vol. 18, no. 7, p. 15501329221113508, Jul. 2022, doi: 10.1177/15501329221113508.

[31] P. Tahmasebi and A. Hezarkhani, “A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation,” *Comput. Geosci.*, vol. 42, pp. 18–27, May 2012, doi: 10.1016/j.cageo.2012.02.004.

[32] MathWorks, “Simulink,” *MathWorks*. https://la.mathworks.com/products/simulink.html (accessed Jun. 13, 2023).

[33] M. Safeea and P. Neto, “Model-based hardware in the loop control of collaborative robots: Simulink and Python based interfaces,” *Int. J. Comput. Integr. Manuf.*, vol. 0, no. 0, pp. 1–13, Feb. 2023, doi: 10.1080/0951192X.2023.2177744.

[34] Coppelia Robotics, “Robot simulator CoppeliaSim: create, compose, simulate, any robot,” *Coppelia Robotics*. https://www.coppeliarobotics.com/ (accessed Jun. 13, 2023).

[35] R. Sudandhira Veeran, S. Suraj, S. Vali, and N. Dhanush, “CoppeliaSim: Adaptable modular robot and its different locomotions simulation framework,” *Mater. Today Proc.*, Feb. 2021, doi: 10.1016/j.matpr.2021.01.055.

[36] E. Pot, J. Monceaux, R. Gelin, and B. Maisonnier, “Choregraphe: a graphical tool for humanoid robot programming,” in *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, Sep. 2009, pp. 46–51. doi: 10.1109/ROMAN.2009.5326209.

[37] R. Gelin, “NAO,” in *Humanoid Robotics: A Reference*, A. Goswami and P. Vadakkepat, Eds., Dordrecht: Springer Netherlands, 2019, pp. 147–168. doi: 10.1007/978-94-007-6046-2_14.

[38] Aldebaran, “Python SDK — Aldebaran 2.5.11.14a documentation,” *Softbank Robotics Documentation*. http://doc.aldebaran.com/2-5/dev/python/index.html (accessed Aug. 15, 2023).

[39] Redacción Tokio, “Main characteristics of Python: find out about them,” *Tokio*, Mar. 03, 2023. https://www.tokioschool.com/en/news/main-characteristics-python/ (accessed Jun. 21, 2023).

[40] Spyder, “Overview - Spyder IDE,” *Spyder*. https://www.spyder-ide.org/ (accessed Jun. 14, 2023).

[41] Aldebaran, “Locomotion control API — Aldebaran 2.5.11.14a documentation,” *Softbank Robotics Documentation*. http://doc.aldebaran.com/2-5/naoqi/motion/control-walk-api.html#ALMotionProxy::move__floatCR.floatCR.floatCR (accessed Aug. 12, 2023).

[42] S. Dafarra *et al.*, “A Control Architecture with Online Predictive Planning for Position and Torque Controlled Walking of Humanoid Robots,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1–9. doi: 10.1109/IROS.2018.8594277.

[43] L.-H. Juang and J.-S. Zhang, “Robust visual line-following navigation system for humanoid robots,” *Artif. Intell. Rev.*, vol. 53, no. 1, pp. 653–670, Jan. 2020, doi: 10.1007/s10462-018-9672-9.

[44] “Lógica Difusa y Sistemas de Control,” Piura, Perú, p. 26. [Online]. Available: http://www.biblioteca.udep.edu.pe/bibvirudep/tesis/pdf/1_185_184_133_1746.pdf

[45] O. Camacho, E. Iglesias, M. Herrera, and H. Aboukheir, “Control basado en lógica difusa: De los fundamentos a las aplicaciones,” *Novasinergia ISSN 2631-2654*, vol. 4, no. 2, Art. no. 2, Dec. 2021, doi: 10.37135/ns.01.08.01.

[46] J. Postel, “User Datagram Protocol,” Internet Engineering Task Force, Request for Comments RFC 768, Aug. 1980. doi: 10.17487/RFC0768.

[47] W. A. Kamil, S. A. Nor, and R. Alubady, “Performance Evaluation of TCP, UDP and DCCP Traffic Over 4G Network,” *Res. J. Appl. Sci. Eng. Technol.*, vol. 11, no. 10, Art. no. 10, 2015, doi: 10.19026/rjaset.11.2118.

[48] “UDP: ¿qué es el protocolo UDP?,” *IONOS Digital Guide*, Jun. 05, 2020. https://www.ionos.es/digitalguide/servidores/know-how/udp-user-datagram-protocol/ (accessed Jul. 15, 2023).

[49] Aldebaran, “ALMotion — Aldebaran 2.5.11.14a documentation,” *Softbank Robotics Documentation*. http://doc.aldebaran.com/2-5/naoqi/motion/almotion.html#almotion (accessed Aug. 12, 2023).

[50] Aldebaran, “Key concepts — Aldebaran 2.5.11.14a documentation,” *Softbank Robotics Documentation*. http://doc.aldebaran.com/2-5/dev/naoqi/index.html#naoqi-proxy (accessed Aug. 12, 2023).

[51] Aldebaran, “Joint control API — Aldebaran 2.5.11.14a documentation,” *Softbank Robotics Documentation*. http://doc.aldebaran.com/2-5/naoqi/motion/control-joint-api.html#ALMotionProxy::getAngles__AL::ALValueCR.bCR (accessed Aug. 12, 2023).

[52] ISO, “ISO 9241-161:2016, Ergonomics of human-system interaction — Part 161: Guidance on visual user-interface elements,” *ISO*. https://www.iso.org/obp/ui/en/#iso:std:iso:9241:-161:ed-1:v1:en (accessed Aug. 13, 2023).

[53] MathWorks, “Fuzzy Logic Toolbox Documentation - MathWorks América Latina,” *Help Center - MathWorks*. https://la.mathworks.com/help/fuzzy/index.html?s_tid=CRUX_lftnav (accessed Aug. 15, 2023).

[54] 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,” Bachelor Thesis, Escuela Politécnica Nacional, Quito, 2021., 2021. Accessed: Aug. 15, 2023. [Online]. Available: http://bibdigital.epn.edu.ec/handle/15000/21500

[55] MathWorks, “Versions of Python Compatible with MATLAB Products by Release,” *MathWorks*. https://la.mathworks.com/support/requirements/python-compatibility.html (accessed Aug. 15, 2023).

5 ANEXOS

ANEXO I. Aplicación de Fuzzy Logic Toolbox de Matlab.

I.1 Introducción

El “Fuzzy Logic Toolbox” de Matlab se encuentra conformado por un grupo de funciones, aplicaciones y bloques de Simulink que permiten el análisis, diseño y simulación de sistemas basados en lógica difusa. Para el presente proyecto, se hizo uso de la aplicación “Fuzzy Logic Designer”, que pertenece al “Fuzzy Logic Toolbox” de Matlab. Esta aplicación permite realizar el diseño y poner a prueba sistemas de inferencia difusos. Por otro lado, en Simulink, se emplea el bloque de función conocido como "Fuzzy Logic Controller", el cual facilita la implementación e integración del sistema de inferencia difusa desarrollado en el "Fuzzy Logic Designer" con un modelo completo de sistema de control en Simulink [53].

I.2 Manual de Usuario para operación del Fuzzy Logic Designer

A continuación, se detalla el proceso de diseño del sistema de inferencia difuso en “Fuzzy Logic Designer”.

Para abrir la aplicación, se debe escribir “fuzzy” o “fuzzyLogicDesigner” en el Command Window de Matlab”. En la Figura 5.1, se muestra la ventana principal del “Fuzzy Logic Designer”, donde se aprecia cada parte que compone la aplicación.

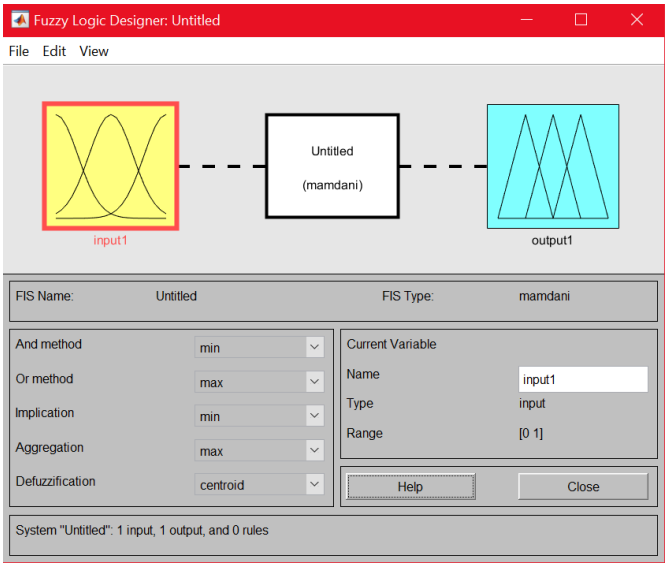


Figura 5.1. Ventana Principal / Ventana de propiedades del Fuzzy Logic Designer.

En la ventana principal del "Fuzzy Logic Designer" se puede apreciar la información sobre las variables de entrada y salida que componen el sistema de inferencia difuso. En el centro

de la ventana, un bloque en blanco refleja el nombre y tipo de sistema de inferencia difuso que está siendo diseñado. Además, se visualizan los operadores asignados a cada método de configuración del sistema difuso. Una exposición detallada de estos métodos se encuentra en la sección 2.3.

Cada uno de estos parámetros y propiedades del sistema difuso puede ser modificado. Se puede editar el nombre de las variables de entrada o salida en el apartado “Current Variable”, “Name”, como se observa en la Figura 5.1. Del mismo modo, se puede agregar o eliminar variables accediendo a ***Edit > Add Variable / Remove Selected Variable***, como se observa en la Figura 5.2.

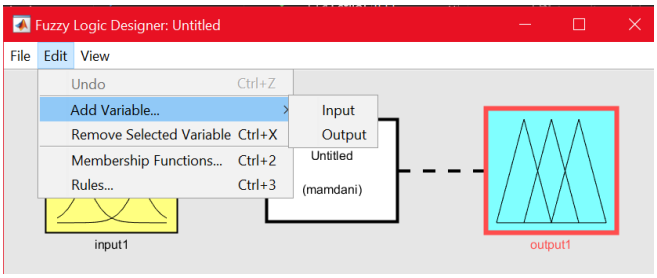


Figura 5.2. Pestaña “Edit” de la ventana principal del Fuzzy Logic Designer.

Haciendo doble clic sobre cualquier variable de entrada o salida se accede a la ventana de edición de las funciones de membresía de la variable correspondiente, como se observa en la Figura 5.3. En esta ventana, se aprecian las funciones de membresía de la variable seleccionada. Además, es posible configurar el nombre (o variable lingüística), el tipo y parámetros de cada función de membresía. Del mismo modo, en “Range” se puede ingresar el universo de discurso de las funciones de membresía en forma de intervalo.

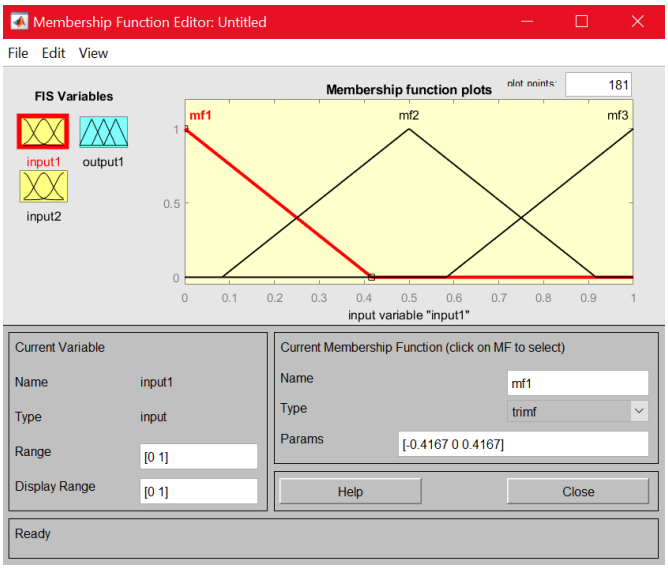


Figura 5.3. Ventana de edición de funciones de membresía.

Se pueden agregar y eliminar funciones de membresía accediendo a la pestaña “Edit”, como se observa en la Figura 5.4.

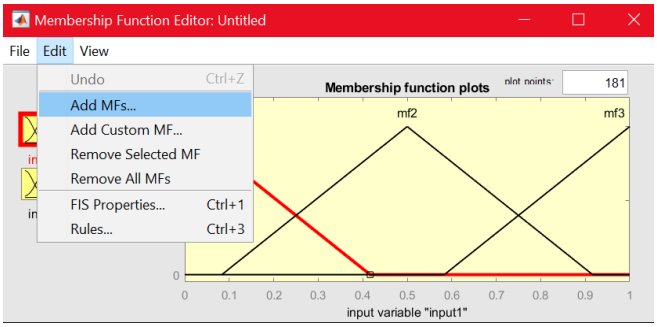


Figura 5.4. Pestaña “Edit” de la ventana de edición de funciones de membresía.

Al dar doble clic en el bloque central de color blanco de la ventana principal de la aplicación “Fuzzy Logic Designer” (Figura 5.1) se accede al editor de reglas del sistema de inferencia difuso, como se aprecia en la Figura 5.5.

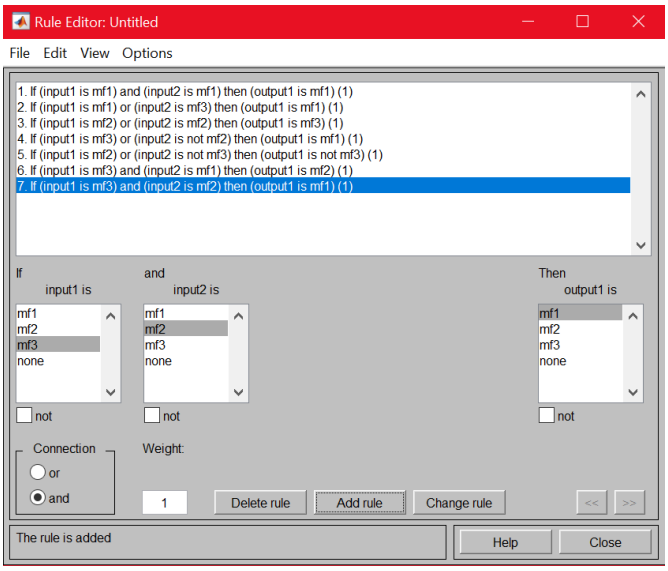
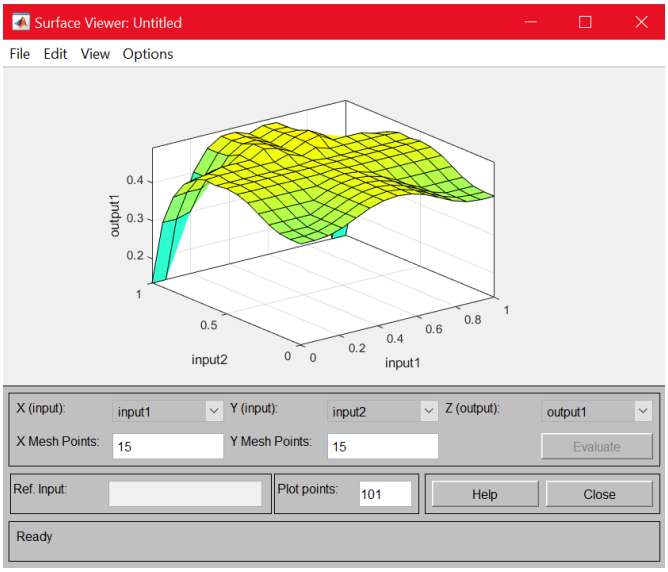


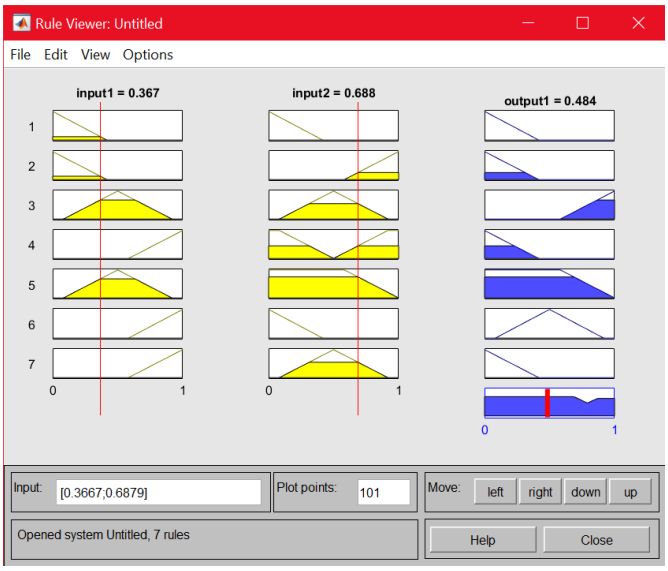
Figura 5.5. Editor de reglas de inferencia difusas.

En el editor de reglas difusas se puede agregar, eliminar o modificar las reglas de inferencia tipo IF – THEN del sistema. Se puede hacer uso de conectores AND, OR y negaciones NOT para establecer las reglas difusas.

Desde cualquier ventana de la aplicación, al dar clic la pestaña “View”, se puede acceder a la superficie difusa (Figura 5.6a) y a la ventana de evaluación de reglas del sistema difuso (Figura 5.6b).



(a)



(b)

Figura 5.6. Ventanas "Viewer" de la aplicación "Fuzzy Logic Designer". (a) Superficie difusa. (b) Evaluación de reglas difusas.

La ventana de evaluación de reglas difusas contiene una interfaz bastante amigable que permite realizar pruebas del funcionamiento del sistema difuso. Estas pruebas consisten en dar valores dentro del universo de discurso a las variables de entrada y obtener un valor en la variable de salida. Esto permite conocer el comportamiento del sistema difuso.

Finalmente, para guardar el sistema de inferencia difuso diseñado, se debe acceder a ***File > Export > To Workspace / To File***. Esto posibilita exportar, ya sea al Workspace de Matlab o a una ubicación específica del explorador de Windows, una estructura *fis* o un

archivo con extensión “.fis”, respectivamente. Es importante tener en cuenta que se puede crear un nuevo sistema difuso desde la pestaña “File”, y escoger el tipo de sistema que se desea diseñar entre “Mamdani” y “Sugeno”, como se observa en la Figura 5.7.

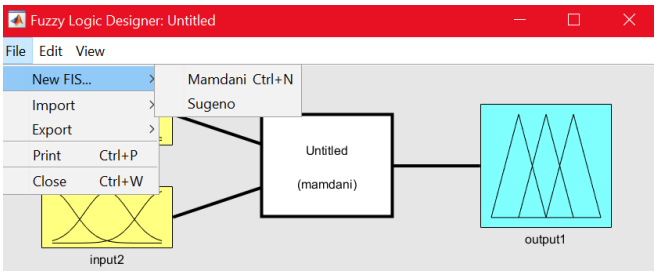


Figura 5.7. Pestaña "File" de la aplicación "Fuzzy Logic Designer".

Para importar una estructura *fis* o un archivo “.fis” a “Fuzzy Logic Designer”, desde el Workspace de Matlab o una ubicación específica del explorador de Windows, respectivamente, se debe acceder a ***File > Import > From Workspace / From File.***

I.2 Manual de Usuario para operación del Fuzzy Logic Controller de Simulink

A continuación, se detalla como enlazar el bloque “Fuzzy Logic Controller” en Simulink con el sistema de inferencia difuso diseñado en el “Fuzzy Logic Designer”.

En Simulink, se debe agregar el bloque “Fuzzy Logic Controller” de la Figura 5.8.

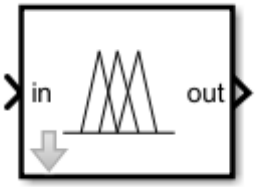


Figura 5.8. Bloque de Simulink "Fuzzy Logic Controller".

Al bloque “Fuzzy Logic Controller” se le conecta en *in* las variables de entrada del sistema difuso diseñado, y en *out*, las variables de salida. En el caso de tener dos o más variables, se debe usar un multiplexor *Mux* de Simulink en el caso de las entradas, y un demultiplexor *Demux* en el caso de las salidas.

Al hacer doble clic sobre el bloque “Fuzzy Logic Controller”, se abre la ventana de parámetros del bloque, como se observa en la Figura 5.9. En esta ventana, se debe escribir el nombre de la estructura *fis* del sistema de inferencia difuso previamente diseñado. Es importante que dicha estructura se encuentre en el Workspace de Matlab para el correcto funcionamiento de la simulación en Simulink.

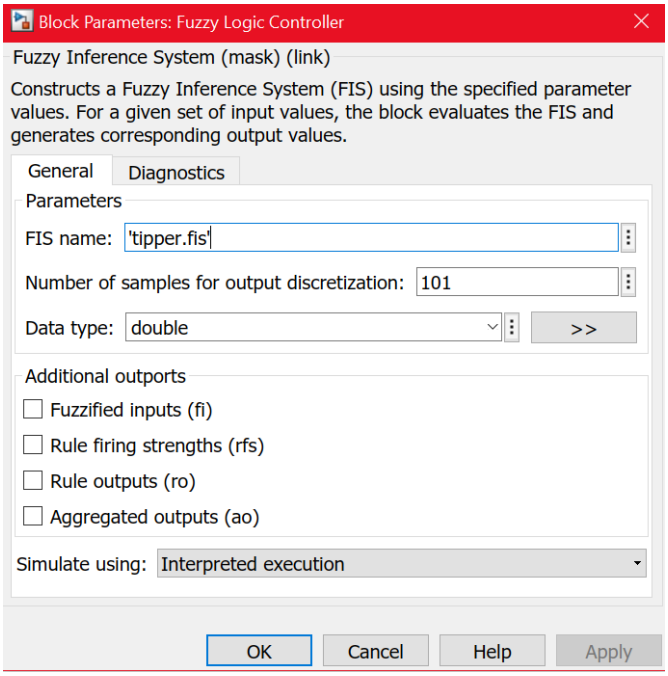


Figura 5.9. Ventana de parámetros del bloque "Fuzzy Logic Controller".

Finalmente, el bloque de Simulink del controlador difuso ya se encuentra enlazado con el sistema de inferencia difuso diseñado en el “Fuzzy Logic Designer”.

I.4 Sistema de control difuso implementado en el proyecto

En el presente proyecto, se diseñó el sistema de inferencia de la Figura 2.16, denominado “Control_FuzzyPD.fis”, y que cuenta con dos variables de entrada y una variable de salida. Cada variable de entrada cuenta con siete funciones de membresía tipo triangular; mientras que la variable de salida contiene cinco funciones de membresía tipo triangular y dos de tipo trapezoidal con el objetivo de ampliar el universo de discurso de la variable de salida y, por ende, la señal de control. El diseño del controlador difuso se lo realiza en la sección 2.6.

En la Figura 5.10, se muestran las ventanas de la aplicación “Fuzzy Logic Designer” donde se diseñó del sistema de controlador difuso empleado en el presente proyecto. Las reglas difusas de la Figura 5.10d fueron programadas de acuerdo con la base de reglas de la Tabla 2.2.

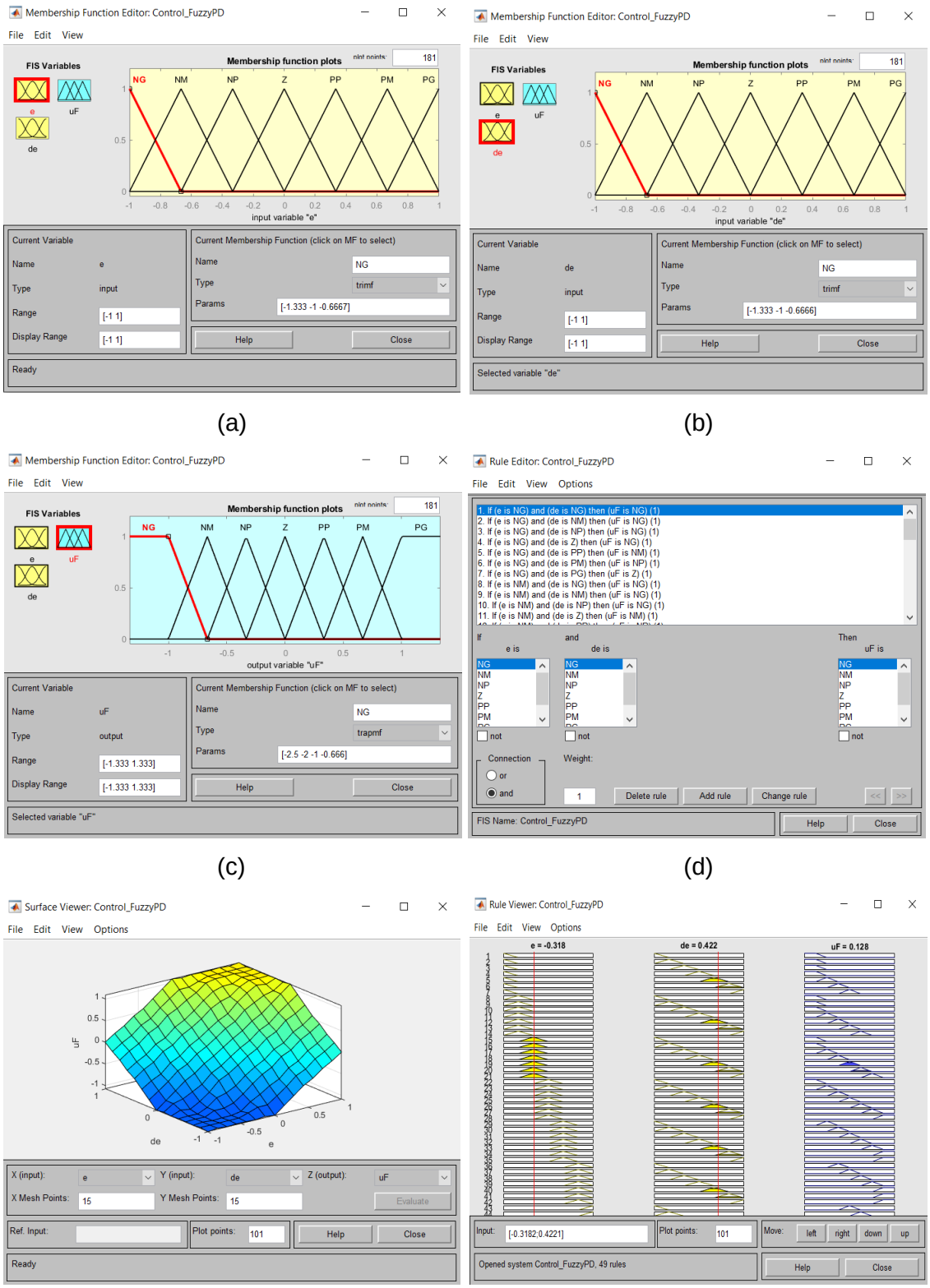


Figura 5.10. Diseño del sistema difuso “Control_FuzzyPD”. (a) Funciones de membresía Variable de entrada “e” de error. (b) Funciones de membresía Variable de entrada “de” de derivada del error. (c) Funciones de membresía Variable de salida “uF” de señal de control. (d) Editor de Reglas. (e) Superficie difusa. (f) Evaluación de reglas difusas.

ANEXO II. Manual de usuario para instalación del sistema.

II.1 Requisitos mínimos

El ordenador precisa de una lista de requerimientos de hardware y software mínimos para ejecutar adecuadamente la simulación del control de seguimiento de trayectorias del robot humanoide NAO. En base a múltiples pruebas, y a [54], se ha determinado los siguientes requisitos mínimos del computador para ejecutar correctamente las simulaciones propuestas en el presente proyecto:

- Procesador Intel Core i7 10ma generación o AMD Ryzen 7.
- Memoria RAM de 16 GB.
- Sistema Operativo Windows 11.
- Tarjeta Gráfica NVIDIA GeForce RTX 2060 6GB.

En base a la lista, se puede notar que el ordenador requiere de elevadas prestaciones computacionales, tanto en procesamiento de datos, memoria y procesamiento de video. En otras palabras, la computadora debe tener un rendimiento elevado y una capacidad de procesamiento significativa para llevar a cabo de manera eficiente la simulación de todo el sistema de control propuesto en este proyecto, asegurando al mismo tiempo la calidad de los resultados. El uso de un ordenador con especificaciones más modestas puede no interferir en la interacción entre los diversos programas requeridos para la simulación. No obstante, al asignarse los recursos informáticos a la comunicación entre estos programas, se produce un impacto en el rendimiento de la simulación, lo cual conlleva a que los resultados obtenidos sean considerablemente distintos de los previstos.

II.2 Instalación del software necesario.

Para la ejecución de la simulación del presente proyecto, se deben instalar todos los programas especificados en la sección 1.4.7. A continuación, se enlistan los programas junto con las versiones utilizadas en el proyecto.

- Anaconda2 – 5.3.0.
- Choregraphe Suite – 2.8.7.4.
- CoppeliaSim Edu – V4.4.0.
- MATLAB / Simulink – R2021b.

Al realizar la instalación de Anaconda2 – 5.3.0 también se instalan la versión de Python 2.7 y Spyder – 3.3.1. Recordando que éste último corresponde al entorno de desarrollo integrado de Python que se empleará en el presente proyecto.

En la carpeta “Instaladores”, dentro de la carpeta de los archivos del proyecto, se tiene el ejecutable de cada software de simulación necesario, a excepción del correspondiente a Matlab. La instalación de cada programa se la realiza manteniendo las consideraciones y configuraciones por defecto de cada instalador.

Se puede instalar Matlab/Simulink normalmente desde la página oficial de MathWorks. Es importante manejar la versión R2022a o cualquier otra anterior, dado que, a partir de la versión R2022b, Matlab ya no es compatible con Python 2.7 [55]. Es fundamental instalar las herramientas “Fuzzy Logic Toolbox”, “Instrument Toolbox” y “Simulink Real-Time” desde Add-Ons, o junto con la instalación de Matlab.

II.3 Instalación de Python SDK

El Python SDK comprende un conjunto de librerías que facilita la incorporación de las funciones y módulos de NAOqi en un entorno de desarrollo integrado de Python. Esto posibilita el control del robot NAO mediante la programación de comandos específicos directamente desde Python. Para la instalación de Python SDK se debe seguir el siguiente procedimiento:

- Dirigirse a la carpeta “Python SDK”, dentro de la carpeta de los archivos del proyecto.
- Copiar las 7 carpetas contenidas en la carpeta “Python SDK” y pegarlas en la siguiente dirección: C:\Users***Nombre de Usuario***\Anaconda2.
- Escribir “Editar variables de entorno del sistema” en la barra de búsqueda de la barra de herramientas de Windows, y abrir dicha aplicación.
- Dentro de la pestaña “Opciones avanzadas”, acceder a “Variables de entorno…”.
- Agregar una nueva variable de usuario, con nombre “PYTHONPATH” y dirección C:\Users***Nombre de Usuario***\Anaconda2\Lib, como se muestra en la Figura 5.11.

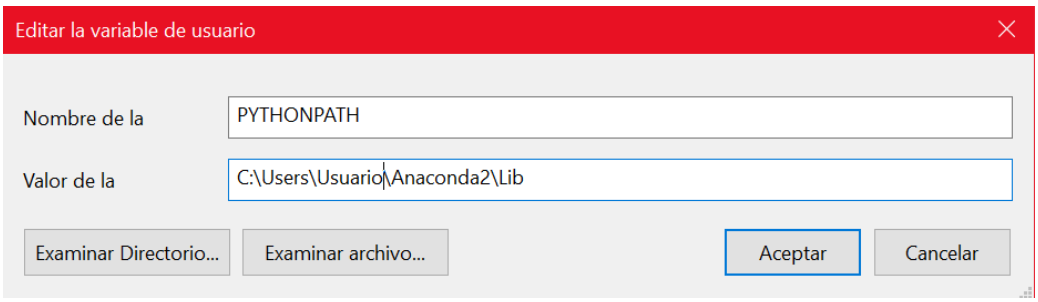


Figura 5.11. Agregar nueva variable de usuario al sistema.

- Clic en “Aceptar”.

II. 4 Configuración de Choregraphe Suite

Es necesario realizar una pequeña configuración inicial después de haber instalado el software Choregraphe con el objetivo de que el modelo de simulación corresponda al robot humanoide NAO V6. Para ello, se realizan los siguientes pasos:

- Abrir Choregraphe Suite.
- Acceder a la pestaña ***Edit > Preferences > Virtual Robot***.
- Escoger "NAO H25 (V6)" dentro de las opciones de "Robot Model", como se observa en la Figura 5.12.

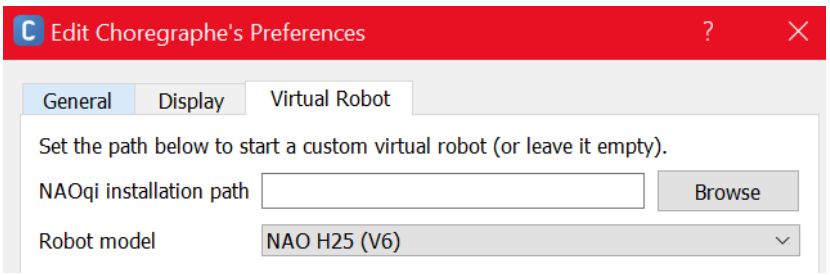


Figura 5.12. Escoger modelo del robot en Choregraphe.

- Clic en "OK".

ANEXO III. Manual de usuario para simulación del sistema.

Para ejecutar la simulación del control de seguimiento de trayectorias del robot humanoide NAO, se debe seguir en orden el siguiente procedimiento:

1. Abrir la interfaz de usuario llamada "HMI", ubicada dentro la carpeta "Interfaz gráfica" de la carpeta de archivos del proyecto. Esto causará que el software Matlab también se inicie.
2. Seguir los pasos que se detallan en la pestaña "Instrucciones" de la interfaz gráfica, los mismos que se detallan a continuación:
   2.1. Abrir los programas: Choregraphe, Matlab, Spyder y CoppeliaSim Edu. En este caso, Matlab ya se encuentra abierto.
   2.2. Realizar la conexión con el robot virtual. Para lo cual, se deben ejecutar los siguientes pasos:
      2.2.1. Acceder a ***Connection > Conect to…*** en Choregraphe.
      2.2.2. Marcar la casilla "Use fixed port: 9559".
      2.2.3. Clic en el robot virtual de la tabla, como se observa en la Figura 5.13. Este robot virtual tiene asignado el nombre del dispositivo Windows.
      2.2.4. Clic en "Select"

2.2.5. En el caso de que la conexión fuera exitosa, se observará, casi inmediatamente, la frase “Connected to ***Host***” en la barra de título de la interfaz de Choregraphe. Donde ***Host*** corresponde al nombre del Host que se muestra en la Figura 5.13. Además, se mostrará el robot virtual en la sección “Robot View” de Choregraphe, como se observa en la Figura 5.14.

2.2.6. En caso de que no se logre establecer una conexión con el robot virtual después de un tiempo considerable tras ejecutar el paso 2.2.4, Choregraphe mostrará una ventana emergente que confirmará la incapacidad de conexión con NAOqi. Ante esta situación, se debe cerrar el software Choregraphe, abrir el Administrador de Tareas del computador y finalizar todas las tareas que contengan cualquiera de estos tres nombres: “naoqi-service”, “qilaunch” y “qi-secure-gateway”. Después de esto, se abre nuevamente Choregraphe y se ejecutan nuevamente los pasos detallados de 2.2.

2.3. Abrir el proyecto de Choregraphe “NAO_virtual.pml”. Para abrir este archivo, acceder a ***File > Open project > “NAO_virtual” > NAO_virtual.pml*** en Choregraphe. Donde “NAO_virtual” es la carpeta del proyecto de Choregraphe y se encuentra en la carpeta “Interfaz gráfica”, entre los archivos del proyecto.

2.4. Abrir el programa “Principal.py” en Spyder. Este programa también se encuentra ubicado en la carpeta “Interfaz gráfica”.

2.5. Abrir la escena “NAO_robot.ttt” en CoppeliaSim Edu. Esta escena también se encuentra ubicada al interior de la carpeta “Interfaz gráfica”.

2.6. En la pestaña “Control Trayectoria” de la interfaz de usuario, se debe escoger la trayectoria de referencia deseada y el tipo de controlador aplicado. Del mismo modo, en esta pestaña, se pueden modificar las ganancias del controlador antes de simular. Los valores de las ganancias que aparecen por defecto en la interfaz de usuario son aquellos con los que se obtuvo los resultados de la sección 3.1.

2.7. Dar clic en el botón “Abrir” de la ventana “Control Trayectoria” de la interfaz de usuario. Esto abrirá el modelo en Simulink del controlador seleccionado y almacenará en el Workspace los datos de las ganancias del controlador, el tiempo de simulación y la estructura *fis* para el funcionamiento del controlador difuso.

2.8. Ejecutar la escena “NAO_robot.ttt”, que había sido abierta previamente en CoppeliaSim Edu.

2.9. Ejecutar el archivo “Principal.py”, que había sido abierto previamente en Spyder.

2.10. Ejecutar la simulación del modelo en Simulink, que se había abierto en el paso 2.7.

2.11. Observar el movimiento del robot NAO en CoppeliaSim Edu.

2.12. Una vez que la simulación del modelo de Simulink haya concluido, detener la ejecución de la escena del robot NAO en CoppeliaSim Edu.

2.13. Verificar que el robot NAO virtual de Choregraphe se haya detenido y se encuentre de pie.

2.13.1. En el caso de que el robot virtual se haya sentado, o continue caminando pese a que la escena de CoppeliaSim Edu haya sido detenida, se debe ejecutar el *behavior* (comportamiento) del proyecto abierto en Choregraphe, “NAO_virtual.pml”. Esto provocará que el robot NAO virtual se detenga, se siente, se levante, y se mantenga en pie.

2.14. Dar clic en el botón “Ejecutar” de la ventana “Control Trayectoria” de la interfaz de usuario.

2.15. Observar los resultados obtenidos del seguimiento de trayectoria en las ventanas “Control Trayectoria” y “Señales” de la interfaz de usuario.

2.16. El robot virtual se encuentra listo para realizar una nueva simulación, para lo cual se debe repetir el procedimiento desde el paso 2.6.

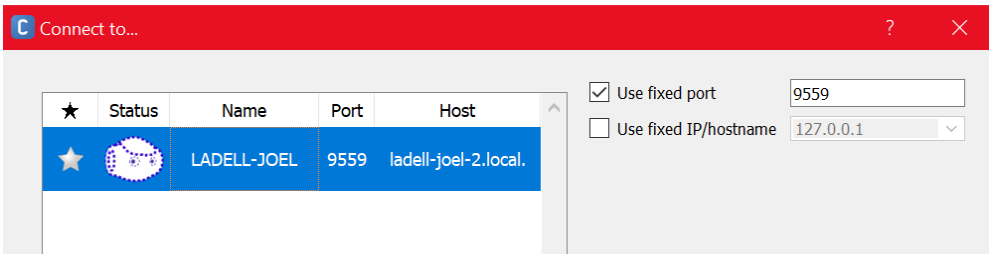


Figura 5.13. Conexión del robot virtual.

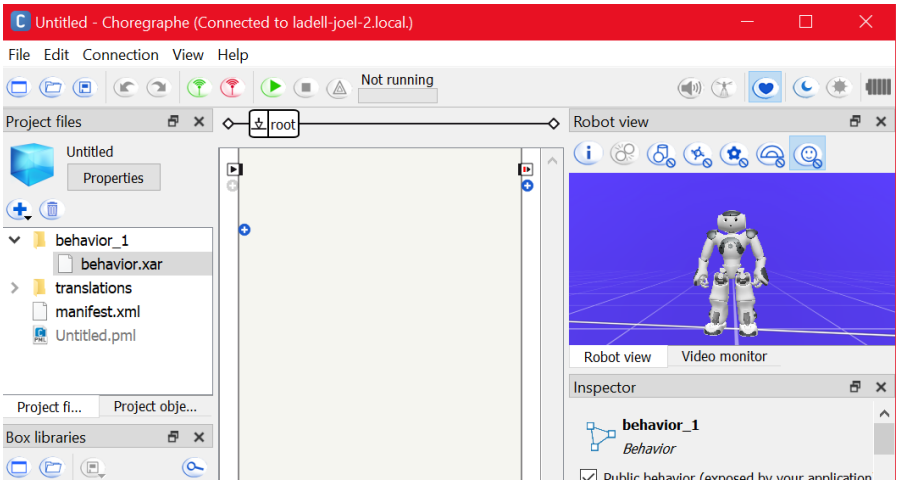


Figura 5.14. Robot virtual conectado en la interfaz de Choregraphe.