ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

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

SIMULACIÓN DE UN CONTROL BASADO EN REDES NEURONALES PARA EL SEGUIMIENTO DE LA TRAYECTORIA DE UN ROBOT HUMANOIDE

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

DARWIN SANTIAGO TRUJILLO BONILLA

darwin.trujillo@epn.edu.ec

DIRECTOR: DR. GEOVANNY DANILO CHÁVEZ GARCÍA

danilo.chavez@epn.edu.ec

DMQ, octubre 2023

CERTIFICACIONES

Yo, DARWIN SANTIAGO TRUJILLO BONILLA 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.

DARWIN SANTIAGO TRUJILLO BONILLA

Certifico que el presente trabajo de integración curricular fue desarrollado por DARWIN SANTIAGO TRUJILLO BONILLA, bajo mi supervisión.

DR. GEOVANNY DANILO CHÁVEZ GARCÍA

DECLARACIÓN DE AUTORÍA

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

DARWIN SANTIAGO TRUJILLO BONILLA

DR. GEOVANNY DANILO CHÁVEZ GARCÍA

DEDICATORIA

“Creo que todos tenemos un poco de
esa bella locura que nos mantiene
andando cuando todo alrededor
es tan insanamente cuerdo.”
-Julio Cortázar

A mi abuelita Elena

AGRADECIMIENTO

A mi familia por su apoyo incondicional para lograr mis metas.

A mis grandes amigos Joelo, Gaby, Sebas, Reno, Tebo, Yi, Richard, David y Pau, quienes han convertido mi vida universitaria en una experiencia llena de momentos divertidos e inolvidables.

Al Dr. Danilo Chávez por ser mi tutor de tesis y permitirme la elaboración de este documento.

Al Dr. Luis Morales e Ing. Nelson Sotomayor por brindarme sus puntos de vista y correcciones en la elaboración de este trabajo.

A todas las personas que se dedican a la investigación. Su labor y dedicación hacia la búsqueda del conocimiento han sido fundamentales para impulsar el progreso y transformar el mundo.

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VII
ABSTRACT VIII
1 INTRODUCCIÓN 1
1.1 Objetivo general 2
1.2 Objetivos específicos 2
1.3 Alcance 3
1.4 Marco teórico 3
1.4.1 Robots Humanoides 3
1.4.2 Robot NAO 4
1.4.3 Librería de NAOqi 6
1.4.4 Cinemática del Robot Móvil 7
1.4.5 Seguimiento de trayectoria 9
1.4.6 Inteligencia Artificial 9
1.4.7 Machine Learning 10
1.4.8 Redes Neuronales 10
1.4.9 Softwares de Simulación 11
1.4.9.1 VREP 11
1.4.9.2 Simulink 12
1.4.9.3 Choregraphe 12
2 METODOLOGÍA 13
2.1 Diseño de Sistemas de Control 13
2.2 Aprendizaje en una Red Neuronal 14
2.2.1 Algoritmos de Machine Learning 14
2.3 Perceptrón Multicapa 15
2.4 Algoritmo de Backpropagation 18
2.5 Diseño del Controlador Cinemático 19
2.5.1 Obtención de la Posición y Orientación del Robot 21
2.6 Diseño del Controlador basado en Redes Neuronales 21

2.7 Desarrollo de las Trayectorias de Referencia 28
2.8 Configuración Entornos de Simulación para el Robot NAO 29
2.8.1 Entorno de Ejecución de PYTHON 29
2.8.2 VREP- PYTHON 29
2.8.3 Librería NAOqi 31
2.8.4 SIMULINK- PYTHON 31
2.8.5 Diagrama de Flujo del Programa Principal 33
2.9 Desarrollo Interfaz Gráfica 34
3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES 37
3.1 Resultados 37
3.2 Conclusiones 45
3.3 Recomendaciones 45
4 REFERENCIAS BIBLIOGRÁFICAS 46

RESUMEN

Este trabajo presenta una técnica de machine learning basada en soft computing para el control de seguimiento de trayectoria de un robot humanoide mediante redes neuronales. Se propone el desarrollo de un controlador cinemático basado en redes neuronales diseñado a partir de la cinemática directa de un robot móvil con el fin de obtener mejor precisión y rendimiento en el seguimiento de trayectorias. El controlador neuronal parte de una sintonización inicial realizada por el diseñador, mientras que en la fase de aplicación mediante el algoritmo de backpropagation, se actualizarán los valores de los pesos de la red neuronal lo cual equivale a las ganancias del controlador. Para la etapa de aprendizaje el algoritmo constantemente evalúa el comportamiento del sistema en lazo cerrado. Los resultados obtenidos serán analizados y comparados con un controlador cinemático convencional, demostrando cualitativa y cuantitativamente que el enfoque propuesto presenta mejoras en términos de rendimiento ante diferentes trayectorias.

PALABRAS CLAVE: inteligencia artificial, redes neuronales, controlador, cinemática, backpropagation, humanoide.

ABSTRACT

This work presents a machine learning technique based on soft computing for the trajectory tracking control of a humanoid robot using neural networks. It is proposed the development of a kinematic controller based on neural networks designed from the direct kinematics of a mobile robot to obtain better accuracy and performance in trajectory tracking. The neural controller starts from an initial tuning performed by the designer, while in the application phase using the backpropagation algorithm, the values of the neural network weights will be updated which is equivalent to the controller gains. For the learning stage the algorithm constantly evaluates the closed-loop behavior of the system. The results obtained will be analyzed and compared with a conventional kinematic controller, demonstrating qualitatively and quantitatively that the proposed approach presents improvements in terms of performance under different trajectories.

KEYWORDS: artificial intelligence, neural networks, controller, kinematics, backpropagation, humanoid.

1 INTRODUCCIÓN

La inteligencia artificial (IA) es uno de los campos más desarrollados y prometedores de la informática y la ciencia modernas. Surgió en la década de 1950 [1] y su uso ha ido creciendo exponencialmente a medida que ha demostrado sus ventajas en diversas aplicaciones. Hoy en día, la inteligencia artificial está transformando rápidamente el mundo y su impacto puede verse en diferentes áreas de investigación, como la robótica [2], la medicina [3], la educación [4] y en aplicaciones como los problemas de clasificación y agrupamiento de elementos [1], sin necesidad de intervención humana constante. Por lo tanto, es necesario conocer sus posibilidades y limitaciones que ofrece la inteligencia artificial para evaluar cuándo es posible utilizarla y cómo aprovecharla eficazmente.

La robótica se ocupa del diseño, la construcción y la operación de los robots. Existen diferentes tipos de robots, entre los que se encuentran los robots móviles y los robots humanoides. Los robos móviles se diseñan y construyen para que puedan moverse y funcionar de manera autónoma [5]. Estos robots pueden utilizarse en aplicaciones como la inspección de infraestructuras [6], la agricultura [7], tareas de acceso a lugares peligrosos para el ser humano [8], seguimiento de trayectorias [9], etc. Por otro lado, los robots humanoides asemejan su forma y estructura al cuerpo humano y están diseñados para moverse y operar en diferentes entornos imitando el comportamiento de un ser humano. Se consideran una de las formas más avanzadas en robótica ya que utilizan tecnología avanzada para tener una experiencia más natural y humana [10]. Al igual que los robots móviles, estos robots se utilizan en áreas de investigación [11], medicina [12] y robótica[11], [13].

El control de seguimiento de trayectoria es una de las tareas más comunes de los robots móviles y humanoides por lo que es un campo de estudio vigente en la actualidad, los controladores se diseñan utilizando modelos matemáticos que representan con gran exactitud el comportamiento del robot. En el caso de los modelos diseñados de robots móviles y robots humanoides, pueden surgir problemas de incertidumbre del modelo y también problemas de no linealidad dependiendo de la complejidad del sistema [14]. Por ello, actualmente se están investigando diferentes técnicas de control inteligente como las redes neuronales artificiales, los algoritmos genéticos, la lógica difusa, etc., con la finalidad de solucionar estas dificultades y otros problemas que puedan surgir.

Las redes neuronales artificiales tienen la característica de que pueden aproximar sistemas o procesos no lineales o de los que no se dispone de mucha información [5]. Por lo tanto, el diseño de un controlador que pueda compensar los parámetros de incertidumbre y la no

linealidad del modelo mejorará la respuesta del robot humanoide, para el seguimiento de la trayectoria.

En el presente trabajo se utilizan redes neuronales para el diseño y autoajuste del controlador. La arquitectura de la red neuronal es similar a la de un modelo perceptrón multicapa el cual mediante el algoritmo de backpropagation minimiza el error de seguimiento de la trayectoria del robot humanoide NAO ajustando las ganancias del controlador con el objetivo de mejorar el rendimiento del robot al seguir una trayectoria deseada. El controlador diseñado no requiere de una fase de aprendizaje off-line ya que inicial su funcionamiento con las ganancias de un control cinemático convencional para mediante su funcionamiento, aprender del comportamiento de la planta y autoajustarlas. Finalmente, se analiza el rendimiento del controlador propuesto a través de índices de desempeño utilizados en sistemas de control y se lo compara con el controlador cinemático sin autoajuste para observar la mejora del comportamiento del sistema ante diferentes trayectorias.

1.1 Objetivo general

Diseñar y simular un controlador basado en redes neuronales 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 y el control de seguimiento de trayectorias, así como sobre inteligencia artificial haciendo énfasis en la técnica de machine learning basada en soft computing.
2. Diseñar e implementar un controlador cinemático convencional y un controlador basado en redes neuronales que permita el seguimiento de trayectoria del robot NAO.
3. Establecer la comunicación entre los programas necesarios para el movimiento del robot NAO.
4. Probar el funcionamiento del controlador desarrollado, verificando que el robot NAO realice el seguimiento de trayectorias.
5. Analizar el desempeño del controlador implementado mediante índices de desempeño utilizados en sistemas de control.

1.3 Alcance

Realizar una revisión bibliográfica sobre los robots humanoides haciendo énfasis en el robot humanoide NAO.

Realizar una revisión bibliográfica del modelo cinemático y control de seguimiento de trayectoria de robots móviles con el propósito de desarrollar un controlador basado en redes neuronales para el control de seguimiento de trayectoria del robot NAO.

Realizar una revisión bibliográfica acerca de machine learning y redes neuronales para realizar un controlador cinemático basado en redes neuronales.

Diseñar e implementar un controlador cinemático convencional y un controlador basado en redes neuronales que permita el control de seguimiento de trayectoria del robot NAO.

Establecer una comunicación entre el entorno de desarrollo integrado de Python, Choregraphe, VREP y Matlab/Simulink, con el propósito de realizar la simulación de movimiento y el control de seguimiento de trayectoria del robot NAO.

Realizar pruebas a nivel de simulación para comprobar el desempeño de los controladores mediante índices de desempeño utilizados en sistemas de control.

1.4 Marco teórico

En este apartado se llevará a cabo una revisión bibliográfica de la propuesta para abordar el problema que se pretende resolver con este proyecto. Es esencial adquirir conocimientos acerca de los robots humanoides, haciendo énfasis en el robot humanoide NAO y sus especificaciones generales, así como comprender el modelo cinemático del robot móvil que se utilizará para calcular las posiciones y velocidades del robot humanoide NAO. Además, es necesario comprender los conceptos de inteligencia artificial y redes neuronales.

1.4.1 Robots Humanoides

Los robots humanoides son robots cuya apariencia se fundamenta en el cuerpo humano. Así, estos robots se modelan con piernas, brazos, manos e incluso pueden tener rostro con ojos y boca [15].

Existen diferentes tipos de robots humanoides [10] que se utilizan tanto en aplicaciones de atención sanitaria [16] como en entornos como viviendas [17], escuelas [18], museos [19] o fuerzas armadas [20]. El uso de estos robots no es exclusivamente para entretenimiento, sino que también brindan ayuda, terapia, comunicación, educación, etc. [21]

Un robot humanoide para que sea útil en la ejecución de tareas, como desplazarse en un entorno de trabajo, debe ser estáticamente estable. Esta condición se cumple si el centro de gravedad se encuentra dentro del polígono formado por los puntos de contacto de sus pies con el suelo [5]. La locomoción bípeda en los robots humanoides necesita una compleja interacción de las características mecánicas y el sistema de control. Se trata de un proceso complejo, no lineal y dinámico que se realiza de tal manera que se producen pequeñas fluctuaciones, que no pertenecen a la trayectoria sino a la medición de datos. El sistema de caminata del humanoide permite establecer dos variables de control: la velocidad lineal u (ejes x, y) y la velocidad angular ω (eje z), estas velocidades serán las entradas del sistema ya que pueden ser ajustables. El objetivo de control del robot es navegar hacia los puntos específicos del plano x-y por lo que los parámetros de posición del robot corresponderán a las salidas del sistema [11].

Los robots humanoides, al seguir la trayectoria, presentan oscilaciones periódicas, esto se debe a que el punto de desplazamiento del robot se encuentra en el pecho y al desplazarse se verá afectado por el movimiento de las piernas del humanoide, generando esta particularidad del movimiento. Esta característica también se refleja en las señales de control del robot [22].

1.4.2 Robot NAO

El robot humanoide utilizado en este trabajo corresponde a la sexta versión comercial del robot humanoide NAO, desarrollado por la empresa francesa Aldebaran-Robotics. La **Figura 1.1** muestra la estructura del robot NAO. Es un robot interactivo, programable y personalizable [23].

Actualmente se utiliza como una interfaz innovadora para proporcionar información a los clientes, interacción con niños autistas y como una herramienta de investigación. Cuenta con una plataforma robótica avanzada que permite realizar tareas de interacción humano-máquina, navegación, desplazamiento o algoritmos de movimiento. Se ha convertido en un ícono en las áreas de la investigación, la educación, la salud y el comercio [23].

Figura 1.1 Robot NAO.

1.4.2.1 Especificaciones Generales

Tiene una altura de 57.4 cm y un peso de 5.48 kg. Su índice de masa corporal (13.5 kg/m^2) es significativamente inferior al de otros humanoides, como ASIMO (28.4 kg/m^2) o HOAP-3 (25 kg/m^2). Dispone de 25 grados de libertad: 5 están situados en cada pierna, 5 en cada brazo, 2 en cada mano y 2 en el cuello. La **Figura 1.2** muestra los distintos grados de libertad que posee el robot humanoide NAO. La última articulación accionada está situada en la pelvis, tiene una inclinación de 45° y está acoplada a la articulación de la cadera de la otra pierna, que también tiene una inclinación de 45°. Esta configuración ahorra un motor en comparación al diseño cinemático de 6 grados de libertad por pierna de otros humanoides y con este diseño mecánico se consigue una amplia gama de movimientos y un fácil mantenimiento [13].

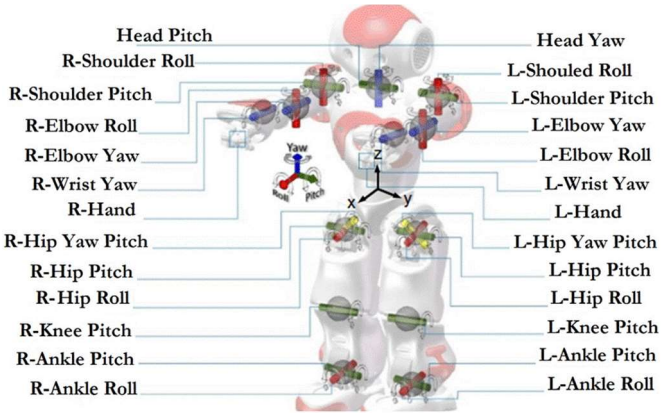


Figura 1.2 Grados de Libertad Robot NAO [24].

En cuanto a sus capacidades informáticas, el robot está equipado con una placa madre ATOM E3845, CPU Quad Core. Cuenta con una cámara de video CMOS de 30 fps con una resolución de 640x480 [15]. El robot se comunica con ordenadores remotos a través de un puerto Ethernet. Dispone de una interfaz gráfica de usuario GUI, con la que se puede controlar el movimiento del robot [25]. Su sistema de actuación está compuesto por motores Brush DC Coreless, conocidos por su precisión y fiabilidad. Está equipado con engranajes rectos de PTFE (politetrafluoroetileno) y engranajes epicicloidales. Con esta estructura el robot puede alcanzar una velocidad máxima de 0.6 km/h [13].

Sus módulos de software integrados, llamados CHOREGRAPHE, permiten la conversión de texto a voz, la localización sonidos, y la detección visual de colores, rostros, patrones y obstáculos. Su sistema operativo NAOqi puede utilizarse en Windows, Mac y Linux, y el desarrollo de aplicaciones mediante lenguajes de programación como C++, Urbi, Python o Java [15] [23].

NAO dispone en su interior un conjunto cuya información se actualiza cada 20 ms. La adquisición de datos en tiempo real se realiza a partir de 2 girómetros y 3 acelerómetros. Para registrar las posiciones reales de las articulaciones incorpora codificadores rotatorios magnéticos (MRE). Para detectar la entrada de fuerza respecto al suelo utiliza sensores sensibles a la fuerza (FSR). Posee unos topes que funcionan como detectores de colisión y sensores capacitivos situados en la frente para recibir entradas táctiles [15].

En la **Figura 1.3** se puede apreciar la distribución de los distintos puertos y sensores presentes en el robot NAO.

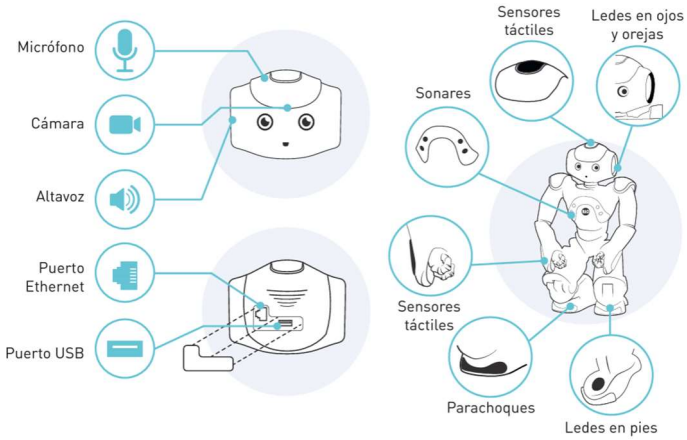


Figura 1.3 Puertos y Sensores del Robot NAO [5].

1.4.3 Librería de NAOqi

NAOqi permite que varios módulos de software se comuniquen entre sí, permitiendo que se ejecuten y controlen los comandos realizados en el lenguaje de programación Python hasta el robot humanoide [26].

En la **Figura 1.4** se observa que la librería NAOqi funciona como una caja negra, lo que significa que no es necesario comprender su funcionamiento interno para poder utilizarla. En particular, esta librería ofrece una API que permite controlar el movimiento del robot, acceder a sus sensores y actuadores, entre otras funcionalidades [27].

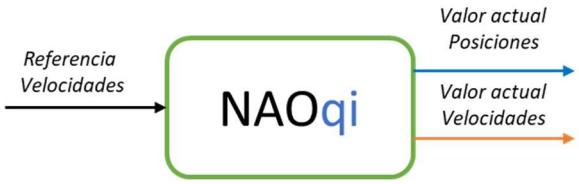


Figura 1.4 Librería NAOqi considerada como una caja negra.

La librería NAOqi y el modelo cinemático de un robot móvil se relacionan indirectamente en la programación del robot NAO. La librería NAOqi se utilizará para enviar las entradas

de control al robot y recibir los datos de los sensores. Dado que la cinemática del robot NAO se representa como una caja negra, el modelo cinemático del robot móvil se utilizará para calcular las posiciones y velocidades del robot en función de esas entradas.

1.4.4 Cinemática del Robot Móvil

El robot móvil diferencial corresponde a un robot móvil terrestre formado por dos ruedas convencionales controladas independientemente y una rueda pasiva para mantener el equilibrio y la estabilidad [28].

Las principales áreas de investigación de este tipo de robot constan en la navegación, la planificación de caminos, la generación de trayectorias, el control de seguimiento de trayectorias y la estabilidad [14]. Se utiliza frecuentemente en sistemas de control debido a su dinámica rápida y no lineal [29].

Para realizar el diseño del controlador para el control del robot móvil, es necesario comprender el comportamiento mecánico del robot. La cinemática se ocupa de la configuración y el comportamiento mecánico de los robots, la relación entre sus parámetros geométricos y las restricciones de movimiento en sus trayectorias. También investiga el movimiento de los cuerpos sin considerar las masas y torques que producen el movimiento. El estudio de la cinemática es fundamental para el estudio del control de trayectorias de los robots [30].

Para el control de trayectorias utilizando robots móviles, generalmente se utilizan controladores diseñados a partir de la cinemática del robot móvil. Sin embargo, para aplicaciones que requieran movimientos con altas velocidades y/o transporte de cargas pesadas, es importante considerar la dinámica del robot, dado que las características dinámicas del robot cambian y para mantener un buen comportamiento es necesario que el controlador se adapte a este tipo de cambios [31].

Como se mencionó anteriormente, para este trabajo se considera el modelo cinemático de un robot móvil para calcular las posiciones y velocidades del robot NAO.

La **Figura 1.5** muestra la geometría de un robot móvil con un punto de control desplazado una distancia a. Donde O es el centro del eje entre las ruedas izquierda y derecha, P es el centro de gravedad del robot móvil y corresponde al punto de control, a es la distancia entre el punto central O que une las ruedas de tracción con el punto P, r es el radio de cada rueda y d es la distancia entre las ruedas, u y ω corresponden a la velocidad lineal y velocidad angular respectivamente, φ es la orientación del robot.

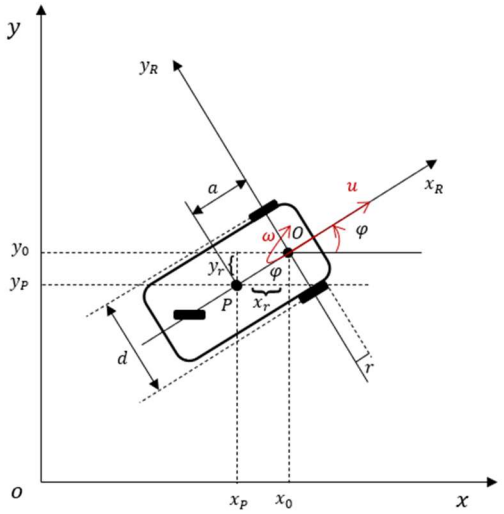


Figura 1.5 Geometría Robot Móvil con Punto de Control Desplazado.

El modelo completo de este robot está dado por [28]

$$\begin{bmatrix} \dot{x}\,(t) \\ \dot{y}\,(t) \\ \dot{\varphi}\,(t) \end{bmatrix} = \begin{bmatrix} \cos\varphi\,(t) & -a\sin\varphi\,(t) \\ \sin\varphi\,(t) & a\cos\varphi\,(t) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u(t) \\ \omega(t) \end{bmatrix} \tag{1.1}$$

La cinemática del robot móvil en el punto P [28], está dada por:

$$\dot{h} = J\,U \tag{1.2}$$

Siendo:

$$\dot{h} = \begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} \tag{1.3}$$

Donde $\dot{h}$ es la referencia de velocidad lineal y angular.

$$J = \begin{bmatrix} \cos\varphi\,(t) & -a\sin\varphi\,(t) \\ \sin\varphi\,(t) & a\cos\varphi\,(t) \end{bmatrix} \tag{1.4}$$

Donde J es la matriz de desacoplamiento. Esta matriz es no singular, dado que $|J| \neq 0$. Por lo que la matriz J es invertible.

$$U = \begin{bmatrix} u(t) \\ \omega(t) \end{bmatrix} \tag{1.5}$$

Obteniendo:

$$\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} = \begin{bmatrix} \cos\varphi\,(t) & -a\sin\varphi\,(t) \\ \sin\varphi\,(t) & a\cos\varphi\,(t) \end{bmatrix} \begin{bmatrix} u(t) \\ \omega(t) \end{bmatrix} \tag{1.6}$$

Con el modelo de la ecuación (1.6) se realizará el diseño del controlador cinemático convencional y a partir del mismo se realizará el diseño del controlador basado en redes neuronales para el seguimiento de trayectoria del robot NAO.

1.4.5 Seguimiento de trayectoria

El control de seguimiento de trayectoria es utilizado en aplicaciones en las que se conoce el entorno de operación. Sus controladores suelen diseñarse a partir de la cinemática del robot y, en los casos en los que se producen cambios en las características dinámicas como la inercia, la masa o el centro de masa, se tiene en cuenta la dinámica del robot [32] [33].

Una vez generada la trayectoria, el robot debe esencialmente seguirla. Es posible que se produzcan errores en el movimiento del robot porque los modelos dinámicos de estos sistemas son complejos de obtener y además puede que existan incertidumbres, perturbaciones, errores de medición de los sensores, etc., que afecten el comportamiento del robot [34]. Por esta razón, el controlador propuesto se aplicará en estos sistemas ya qua aprenderá de la cinemática desconocida del sistema sin la necesidad de utilizar una modelo cinemática del robot, permitiendo que el robot alcance y siga la referencia parametrizada en un valor de tiempo establecido, con el mínimo error de seguimiento en un entorno de simulación de robots [29].

El seguimiento de trayectorias utilizando inteligencia artificial, permite al robot seguir una trayectoria utilizando algoritmos de aprendizaje supervisado, no supervisado, que permiten al robot aprender a partir de la interacción con el entorno [35].

1.4.6 Inteligencia Artificial

El objetivo de la Inteligencia Artificial (IA) es construir sistemas inteligentes como ordenadores, máquinas y otros artefactos que repliquen la inteligencia evidente en los seres humanos, caracterizada por capacidades cognitivas, aprendizaje y toma de decisiones. Con esto el sistema inteligente es capaz de adaptarse a nuevas situaciones, resolver problemas y realizar otras tareas [36].

En la actualidad, se están investigando técnicas de control inteligente, como el soft computing, que incluyen redes neuronales, algoritmos genéticos, lógica difusa, entre otros, para solucionar problemas de robótica móvil. Uno de los principales retos es la navegación automática en entornos con características impredecibles y poca información disponible, donde estas técnicas pueden ser especialmente útiles para lidiar con la incertidumbre del entorno [37], [38].

1.4.7 Machine Learning

Machine Learning (ML) es una técnica de la inteligencia artificial que permite a las máquinas aprender a partir de la experiencia y mejorar su rendimiento en una tarea específica sin necesidad de una programación explícita. Las máquinas aprenden a partir de los datos proporcionados, y utilizan ese conocimiento para tomar decisiones y realizar predicciones en nuevos datos. El proceso de aprendizaje se puede realizar a partir de la implementación de algoritmos de aprendizaje supervisados o no supervisados [39].

1.4.8 Redes Neuronales

Las redes neuronales son una técnica de machine learning basada en soft computing y tienen la capacidad de aproximar sistemas o procesos no lineales. Sus esquemas adaptativos pueden reducir la incertidumbre de sistemas no lineales [40] mediante el ajuste de sus parámetros internos. Las redes neuronales se utilizan generalmente para el reconocimiento de patrones, el control de sistemas y la visión artificial. La red neuronal más utilizada para el control de sistemas es el perceptrón multicapa [5][14].

Las neuronas de las redes neuronales artificiales corresponden a elementos que reciben señales de otras neuronas y se caracterizan por un estado de activación, unos parámetros internos denominados pesos sinápticos y una ecuación de estado que dependerá del modelo escogido. A través de la sinapsis o interconexión se propagan los valores numéricos de una neurona a otra que se evalúan mediante los pesos sinápticos de las conexiones. En las redes neuronales artificiales, los pesos sinápticos se actualizan en intervalos de tiempo preestablecidos. Los pesos sinápticos pueden actualizarse a través del proceso de aprendizaje de la red neuronal [5] [1].

La **Figura 1.6** muestra el modelo de un perceptrón simple. Este modelo es capaz de realizar tareas de clasificación de forma automática. Su arquitectura es sencilla, ya que consiste en una estructura monocapa en la que todas las entradas están conectadas a una neurona que proporciona una única salida binaria. Tiene n ramas de conexiones de entrada para la sinapsis, un nodo sumador y una función de activación. Cada rama tiene un peso w, si éste es positivo cumple con un rol de excitación y si es negativo tiene un rol de inhibición. El nodo suma las señales de entrada multiplicadas por los pesos sinápticos. La función de activación f satura el valor de la señal de salida. El umbral θ corresponde a un factor de comparación para obtener la salida [1].

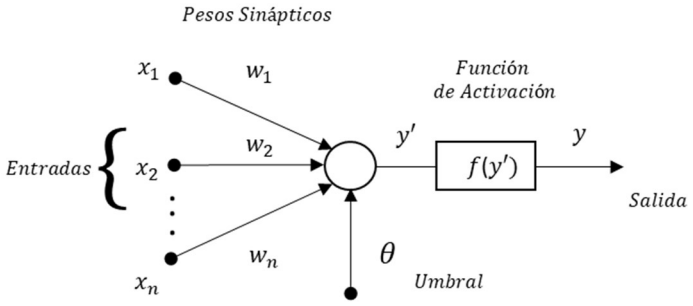


Figura 1.6 Modelo de McCuloch-Pitts o Perceptrón Simple.

La activación de la célula de salida se calcula a partir de la siguiente ecuación:

$$y' = \sum_{i=1}^{n} w_i x_i \tag{1.7}$$

Para el aprendizaje, el perceptrón utiliza la función umbral. La salida final se obtiene aplicando la función de activación en la ecuación (1.7) y sumando el término θ:

$$y' = f\left(\sum_{i=1}^{n} w_i x_i + \theta\right) \tag{1.8}$$

Donde:

$$f(s) = \begin{cases} 1 & si\ s > 0 \\ -1 & en\ caso\ contrario \end{cases}$$

La salida que obtenida de f es binaria, lo que permite realizar una clasificación de las siguientes clases: Si la salida es 1, la entrada es de clase A. Si la salida es -1, la entrada es de clase B.

1.4.9 Softwares de Simulación

A continuación, se detallarán los softwares necesarios para que el robot humanoide envíe y reciba datos correctamente, de forma que se pueda visualizar la simulación del seguimiento de trayectoria del robot.

1.4.9.1 VREP

VREP es un simulador robótico con un entorno de desarrollo integrado el cual es utilizado generalmente en aplicaciones de simulación de sistemas, control de hardware, monitorización remota, presentación de productos, desarrollo de algoritmos de control y robótica [41]. Es eficaz y versátil para simular sistemas robóticos debido a que cada componente del sistema puede ser controlado mediante diferentes mecanismos de control considerando su cinemática, dinámica y el entorno [42].

Los controladores pueden programarse en C++, Python, Matlab/Simulink e integrase fácilmente en este software mediante plug-ins [41].

1.4.9.2 Simulink

Es un entorno de simulación donde se programa mediante diagramas de bloques de multidominio. Es capaz de enviar información a entornos de desarrollo interactivos para Python tal como Spyder. En la **Figura 1.7** se muestran los bloques necesarios para realizar el envío y recepción de datos entre Simulink y el entorno de desarrollo para Python. En estos bloques se debe configurar una dirección IP remota, un puerto remoto y el tiempo de muestreo [43].

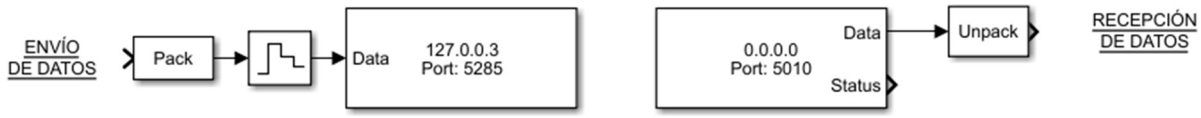


Figura 1.7 Bloque UDP de envío y recepción de datos.

1.4.9.3 Choregraphe

Choregraphe corresponde a un entorno gráfico que permite la programación del NAO. En la **Figura 1.8** se puede apreciar la interfaz de Choregraphe. Cuando Choregraphe se ejecuta en un ordenador, da acceso a las funciones que ofrece Naoqi a través su interfaz gráfica de usuario. La eficacia y el rendimiento de Choregraphe se debe a que Naoqi se ejecuta detrás de Choregraphe. En realidad, Choregraphe es un módulo específico del Naoqi encargado de la representación gráfica para el usuario [26].

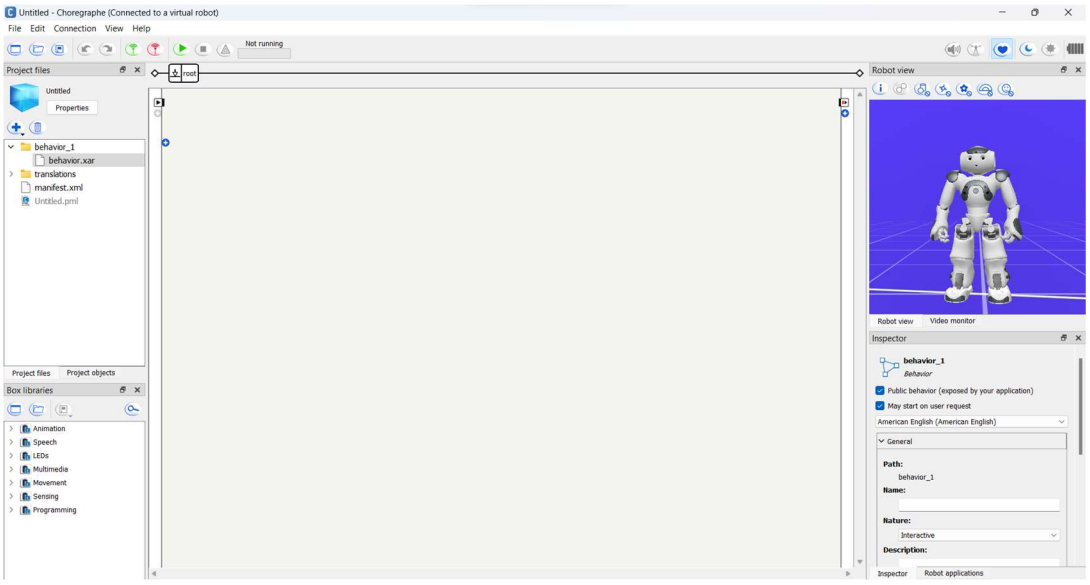


Figura 1.8 Interfaz de Choregraphe.

2 METODOLOGÍA

Este proyecto tiene como objetivo mejorar el rendimiento y precisión del seguimiento de trayectorias de un robot NAO mediante el desarrollo de un controlador cinemático basado en redes neuronales, construidas a partir del modelo cinemático de un robot móvil. Para lograr esto, se ha comparado el enfoque propuesto con un controlador cinemático convencional, utilizando diferentes trayectorias. Los resultados del estudio demuestran que el enfoque basado en redes neuronales supera al enfoque tradicional en términos de rendimiento y precisión en el seguimiento de trayectorias. Estas afirmaciones se respaldan con pruebas cualitativas y cuantitativas presentadas y analizadas en el estudio.

El proyecto sigue una metodología de investigación dividida en cuatro fases. En el capítulo 1, se presenta la fase teórica, donde se realiza una breve revisión bibliográfica de los robots humanoides, haciendo énfasis en el robot NAO, el modelo cinemático del robot móvil, control de seguimiento de trayectoria, así como sobre inteligencia artificial y redes neuronales. El objetivo de esta fase es desarrollar un controlador basado en redes neuronales para el control de seguimiento de trayectoria del robot NAO. En el capítulo 2, se presenta la fase de diseño, donde se muestran la realización de un controlador cinemático convencional y un controlador basado en redes neuronales para el control de seguimiento de trayectorias del robot NAO. En la fase de implementación, se explica el proceso de comunicación entre diferentes programas para llevar a cabo la simulación del movimiento y control del robot. Finalmente, en el capítulo 3, se presenta la fase de pruebas de funcionamiento, donde se realizan pruebas a nivel de simulación para comprobar el desempeño de los controladores mediante índices de desempeño como IAE, ISE, ISU, utilizados en sistemas de control.

2.1 Diseño de Sistemas de Control

Los sistemas de control se diseñan a partir de modelos matemáticos que reflejan información aproximada sobre el funcionamiento de los sistemas. En consecuencia, los modelos realizan simulaciones aproximadas del comportamiento real del sistema [11].

Existen tres categorías principales de modelización:

- Modelización analítica. - Los modelos se calculan mediante ecuaciones diferenciales del sistema que describen su comportamiento [11].
- System Identification. - El proceso de modelado se realiza midiendo los datos de entrada-salida del sistema [11].

- Soft Computing. - Se enfoca en la simulación de la inteligencia humana para modelar la incertidumbre e imprecisión de datos en la toma de decisiones. Como técnicas de esta categoría, podemos incluir a las redes neuronales, los sistemas difusos, los algoritmos evolutivos [11].

2.2 Aprendizaje en una Red Neuronal

El proceso de aprendizaje consiste en modificar los valores de los pesos sinápticos. Esto se consigue minimizando una determinada función objetivo mediante el algoritmo del gradiente descendente estocástico. Deben cumplirse ciertos criterios de convergencia para que los pesos vuelvas a modificarse [5] [1].

Para cumplir con el periodo de aprendizaje, se determina:

- Cuántas veces se introducirán datos a partir de un número fijo de ciclos. Cuando se supere este número, se detiene el aprendizaje [5].
- Una función objetivo para que el proceso de aprendizaje se detenga cuando el valor del error sea inferior a un valor predefinido. En los casos que no se encuentre una solución, deberá elegirse un número de ciclos [5].
- Cuando los valores de los pesos dejan de variar, la red ha convergido y el proceso de aprendizaje se detiene [5].

2.2.1 Algoritmos de Machine Learning

En ML se pueden considerar los siguientes tipos de algoritmos:

- Aprendizaje supervisado. – Los datos de salidas correctos para un conjunto de datos de entrada se conocen previamente. Estos datos son suministrados de forma semiautomática o por un supervisor. Una variación de este tipo de aprendizaje es el aprendizaje por refuerzo, en el que se determina si la salida de ese patrón es adecuada o inadecuada [44].
- Aprendizaje no supervisado. - No requiere que un supervisor se encargue del aprendizaje. Para un conjunto de datos de entrada, no se conocen los datos de salida. Por lo tanto, el objetivo de este aprendizaje es encontrar patrones de similitud, asociación o distinción en la entrada [44].

La **Figura 2.1** y **Figura 2.2** muestran la estructura de estos tipos de aprendizajes.

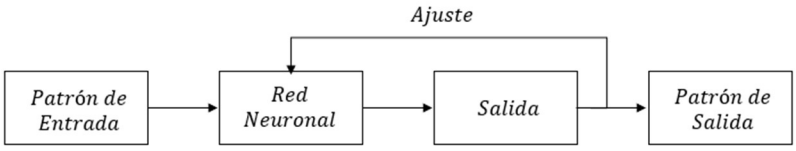


Figura 2.1 Aprendizaje Supervisado.

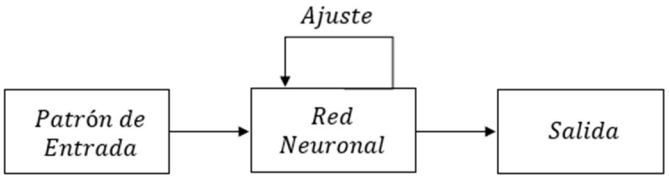


Figura 2.2 Aprendizaje No Supervisado.

2.3 Perceptrón Multicapa

Corresponde a una generalización del perceptrón simple y es uno de los esquemas más utilizados en la actualidad porque es capaz de aproximar sistemas no lineales, filtrar ruido, aproximador universal, etc. Tiene limitaciones debido a la dificultad de realizar un análisis teórico de la red por sus componentes no lineales. Se caracteriza porque las neuronas se agrupan en varios niveles [1].

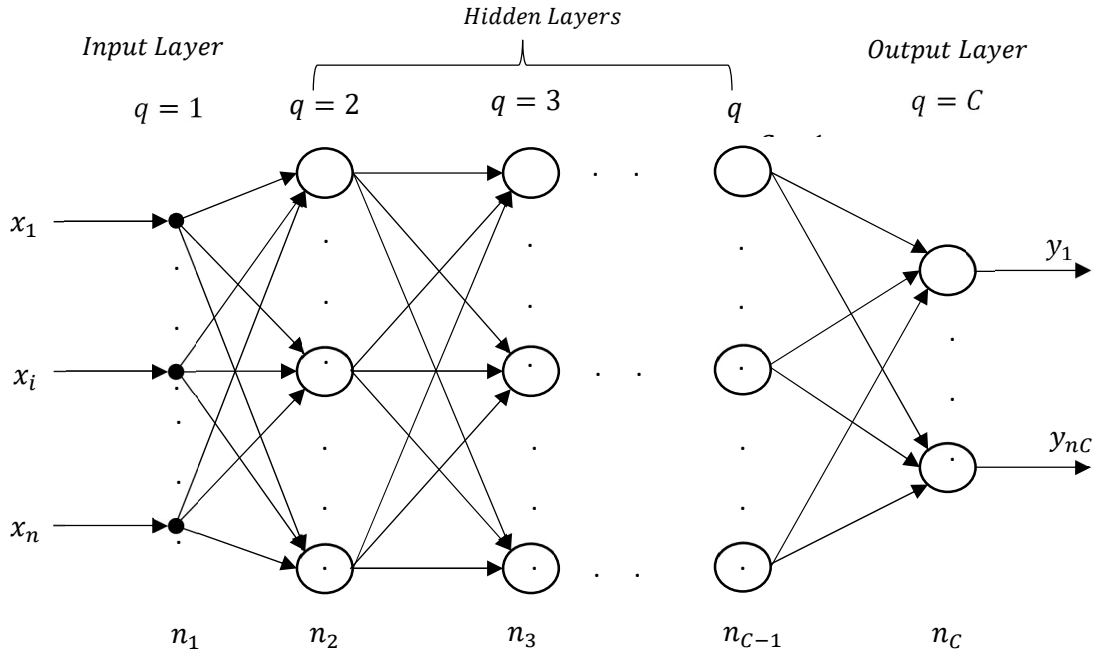


Figura 2.3 Modelo Perceptrón Multicapa.

La **Figura 2.3** muestra el modelo de un perceptrón multicapa. Como en el perceptrón simple, cada interconexión tiene unos valores llamados pesos sinápticos. Se distinguen tres tipos de capas. La capa de entrada recibe las señales de otras neuronas o del exterior y las propaga a las neuronas capa siguiente. En las capas ocultas, las neuronas realizan un procesamiento no lineal de los patrones recibidos. En la última capa, la red da una respuesta a los valores de entrada introducidos. Normalmente, las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente, por lo que hay conectividad en toda la red. Es posible tener redes en las que algunas neuronas de ciertas capas no estén conectadas a neuronas de la capa siguiente, por lo que el peso de la conexión se considera cero. Hay que tener en cuenta que no es posible demostrar que se puedan obtener mejores resultados añadiendo o eliminando conexiones de una capa o capas y también que puede haber tantos umbrales como células de salida existan [1].

Para un perceptrón multicapa con C capas, $C-2$ capas ocultas y $q = 1,2,\ldots,C$.

- La activación de las neuronas en la capa de entrada viene dada por:

$$a_i^1 = x_i \qquad \forall\, i = 1,\ 2\ ,..\, n_1 \tag{2.1}$$

Donde a_i^1 es la activación de la neurona i en la capa de entrada, x_i es el vector de entrada de la red neuronal artificial, n_1 son las neuronas en la capa de entrada.

- La activación de las neuronas de la capa oculta q, viene dada por:

$$a_i^q = f\left(\sum_{j=1}^{n_{q-1}} w_{ji}^{q-1} a_j^{q-1} + \theta_i^q\right) \qquad \forall\, i = 1,\ 2\ ,..\, n_q \text{ ^ } q = 2,3,\ldots,C-1 \tag{2.2}$$

Donde a_i^q es la activación de la neurona i en la capa q, a_j^{q-1} son las activaciones de las neuronas j en la capa $q-1$, n_q son las neuronas en la capa q, w_{ji}^{q-1} es el vector de pesos de la neurona j a la neurona i entre la capa $q-1$ y la capa q, θ_i^qes el vector de umbrales en la capa q.

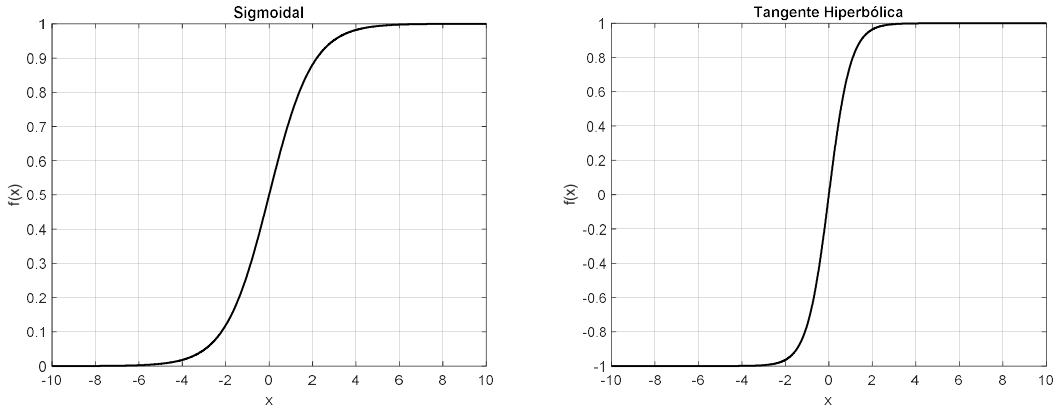
- La activación de las neuronas de la capa de salida a_i^C $(q = C)$, viene dada por:

$$y_i = a_i^C = f\left(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} a_j^{C-1} + \theta_i^C\right) \tag{2.3}$$

$$\forall\, i = 1,\, 2\,, .. n_C$$

Donde y_i es el vector de salida de la red neuronal artificial, a_i^C es la activación de la neurona i en la capa C, a_j^{C-1} son las activaciones de las neuronas j en la capa $C-1$, n_C son las neuronas en la capa C, w_{ji}^{C-1} es el vector de pesos de la neurona j a la neurona i entre la capa $C-1$ y la capa C, θ_i^C es el vector de umbrales en la capa C.

En (2.2) y (2.3) la función de activación f es la responsable de la activación de las neuronas. Esta función puede ser del tipo sigmoidal con valores en el intervalo $[0,1]$ o tangente hiperbólica en el intervalo $[-1,1]$.



(a) (b)

Figura 2.4 Funciones de Activación. (a) Sigmoidal (b) Tangente Hiperbólica.

- Función sigmoidal y su derivada:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

- Derivada de la función sigmoidal:

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(x)(1-\sigma(x)) \tag{2.5}$$

- Función tangente hiperbólica y su derivada:

$$f(x) = \tanh(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{2.6}$$

- Derivada de la tangente hiperbólica:

$$f'(x) = \operatorname{sech}^2(x) = 1 - \tanh^2(x) \tag{2.7}$$

En la **Figura 2.4** se muestra el aspecto de estas funciones de activación. El uso de una u otra función dependerá de los valores de activación de las neuronas que se quieran conseguir.

Las derivadas de estas funciones de activación se utilizarán en la siguiente sección.

2.4 Algoritmo de Backpropagation

Es un algoritmo de aprendizaje supervisado encargado de adaptar los pesos sinápticos para minimizar el error cuadrático medio entre la salida deseada y la salida real. Los errores se propagan en las capas ocultas y en la capa de salida [5].

Por lo tanto, el aprendizaje se plantea como un problema de minimización de la forma:

$$Min_W E \tag{2.8}$$

Donde W son los pesos y umbrales de la red; E es la función del error.

La función del error está descrita como:

$$E(n) = \frac{1}{N}\sum_{n=1}^{N} e(n) \tag{2.9}$$

Donde N es el número de muestras y $e(n)$ es el error cometido por la red para el patrón n, expresado como:

$$e(n) = \frac{1}{2}\sum_{i=1}^{n_C} (s_i(n) - y_i(n))^2 \tag{2.10}$$

Siendo $s_i(n)$ el valor de salida deseada para el patrón n, $y_i(n)$ el valor de salida de la red.

Para encontrar el mínimo de la ecuación (2.9), se utiliza el algoritmo de backpropagation, en donde el error cometido por la red se propaga hacia atrás. Esta regla consiste en modificar cada parámetro w para la minimizar los errores cometidos por la red para cada patrón de entrada n [1].

La ley de aprendizaje es la siguiente:

$$w(n) = w(n-1) - \alpha \frac{\partial e(n)}{\partial w} \tag{2.11}$$

Donde α es la taza de aprendizaje y corresponde al porcentaje de cambio en el que se actualiza cada peso en cada, en un rango de 0 a 1. La forma de obtener el valor adecuado de taza de aprendizaje es heurísticamente [45]. Sin embargo, se puede considerar que con valores grandes de α puede que no se encuentre un mínimo local, y que el algoritmo ni siquiera converja, mientras que con un α puede que converja al mínimo local, pero a costa de un mayor tiempo de procesamiento.

La generalización del algoritmo de backpropagation está descrito como:

$$\begin{gathered} w_{kj}^{C}(n) = w_{kj}^{C}(n-1) - \alpha\, \delta_i^{C-1}(n)\, a_k^{C}(n) \\ \forall\, k = 1,2,\dots,n_c\,;\quad j = 1,2,\dots,n_{c-1}\,;\quad C = 1,2,\dots,C-2 \end{gathered} \tag{2.12}$$

Donde $a_k^C(n)$ es la activación de la neurona k de la capa C para el patrón n y definiendo al término δ asociado a la neurona i de la capa $C-1$ para el patrón n, como:

$$\delta_i^{C-1}(n) = f'\left(\sum_{k=1}^{n_C} w_{kj}^{C} a_k^{C} + \theta_j^{C}\right) \sum_{i=1}^{n_{C-1}} \delta_i^{C-2}(n)\; w_{ji}^{C} \tag{2.13}$$

Donde w_{kj}^C es el vector de pesos de la neurona k a la neurona j entre la capa C y la capa $C-1$.

La ecuación (2.13) necesita la derivada de la función de activación. Como se señaló anteriormente, esta derivada puede ser sigmoidal o tangente hiperbólica y viene dada por las ecuaciones (2.5) y (2.7) respectivamente.

La generalización para los umbrales de la red está descrita como:

$$\begin{gathered} \theta_j^{C-1}(n) = \theta_j^{C-1}(n-1) - \alpha\, \delta_i^{C-1}(n) \\ \forall\; j = 2,\dots,n_{c-1};\quad C = 2,\dots,C-2 \end{gathered} \tag{2.14}$$

Donde θ_j^{C-1}es el vector de umbrales en la capa $C-1$ para el patrón n.

2.5 Diseño del Controlador Cinemático

El algoritmo de control se debe realizar a partir de su posición actual, hasta su posición deseada de llegada.

$$\dot{h_d} = \begin{bmatrix} \dot{x_d}(t) \\ \dot{y_d}(t) \end{bmatrix} \tag{2.15}$$

$$\dot{h_d} = J\ U \tag{2.16}$$

Donde $\dot{h_d}$ corresponde a la referencia de la velocidad lineal y angular. Esta referencia se obtiene a partir las coordenadas deseadas $x_d(t), y_d(t)$ que el robot debe alcanzar.

El objetivo de control es encontrar u y ω para que la posición deseada sea alcanzada en un intervalo de tiempo finito. Teniendo en cuenta las coordenadas del punto de interés $[x, y]^T$.

Por lo tanto, se despeja la ecuación (2.16) y se propone un controlador tipo P descrito por:

$$U = J^{-1}\left[\dot{h_d} + u_c\right] \tag{2.17}$$

Donde u_c es el controlador tipo P dado por:

$$u_c = k\ e(t) \tag{2.18}$$

$e(t)$ corresponde al error de seguimiento y se define como:

$$e(t) = \begin{bmatrix} e_x(t) \\ e_y(t) \end{bmatrix} = \begin{bmatrix} x_d(t) - x(t) \\ y_d(t) - y(t) \end{bmatrix} \tag{2.19}$$

Donde $e_x(t)$ y $e_y(t)$ son los errores de posición en los ejes X e Y respectivamente, k_p es la matriz de ganancias:

$$k_p = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix} \tag{2.20}$$

Siendo $k_x > 0$, $k_y > 0$, los valores de las ganancias del controlador y se obtienen heurísticamente [46].

Reemplazando las ecuaciones (2.18), (2.19) y (2.20) en la ecuación (2.17) se tiene la ley de control para el control de trayectorias es:

$$\begin{bmatrix} u_{ref}(t) \\ \omega_{ref}(t) \end{bmatrix} = \begin{bmatrix} \cos\varphi\,(t) & \sin\varphi\,(t) \\ -\frac{1}{a}\sin\varphi\,(t) & \frac{1}{a}\cos\varphi\,(t) \end{bmatrix} \begin{bmatrix} \dot{x_d}(t) + k_x\ e_x(t) \\ \dot{y_d}(t) + k_y\ e_y(t) \end{bmatrix} \tag{2.21}$$

Donde $\left[u_{ref}(t), \omega_{ref}(t)\right]^T$ es la salida del controlador cinemático. Para el diseño del controlador cinemático se considera que: $k_x = k_p\ y\ \ k_y = k_p$

2.5.1 Obtención de la Posición y Orientación del Robot

La posición y orientación del robot se obtiene integrando los valores de la ecuación (1.1) en un periodo de tiempo Δt [14]:

$$x(t) = x(t_0) + \int_{t_0}^{t_f} (u(t) * \cos\varphi\,(t) - a\,\omega(t) \sin\varphi\,(t))\,dt \tag{2.22}$$

$$y(t) = y(t_0) + \int_{t_0}^{t_f} (u(t) * \sin\varphi\,(t) + a\,\omega(t) \cos\varphi\,(t)\,)dt \tag{2.23}$$

$$\varphi(t) = \varphi(t_0) + \int_{t_0}^{t_f} \omega(t)\,dt \tag{2.24}$$

Asumiendo una configuración del robot $[x_k\ y_k\ \varphi_k]$ y que las entradas de velocidad u_k y ω_k son conocidas en el tiempo discreto t_k, entonces utilizando el método de integración de Euler [47] en las ecuaciones (2.22),(2.23) y (2.24), se tiene que:

$$x_{k+1} = x_k + T_s * (u_k\ \cos\varphi_k - a\ \omega_k \sin\varphi_k) \tag{2.25}$$

$$y_{k+1} = y_k + T_s * (u_k\ \sin\varphi_k + a\ \omega_k \cos\varphi_k) \tag{2.26}$$

$$\varphi_{k+1} = \varphi_k + \omega_k * T_s \tag{2.27}$$

Reemplazando $\Delta S = u_k T_s$, $\Delta\varphi = \omega_k T_s$ y $T_s = t_{k+1} - t_k$ en las ecuaciones (2.25),(2.26) y (2.27), la estimación de la configuración del robot al tiempo t_k de manera matricial se calcula como:

$$\begin{bmatrix} x_k \\ y_k \\ \varphi_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \varphi_{k-1} \end{bmatrix} + \begin{bmatrix} \cos\varphi_k & -a\sin\varphi_k \\ \sin\varphi_k & a\cos\varphi_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta S \\ \Delta\varphi \end{bmatrix} \tag{2.28}$$

2.6 Diseño del Controlador basado en Redes Neuronales

Los sistemas clásicos de control de seguimiento para robots móviles no compensan la incertidumbre del modelo, la variación de los parámetros de la planta y las perturbaciones externas. Un controlador eficaz debe producir un comportamiento adecuado incluso en ausencia de estas compensaciones [14].

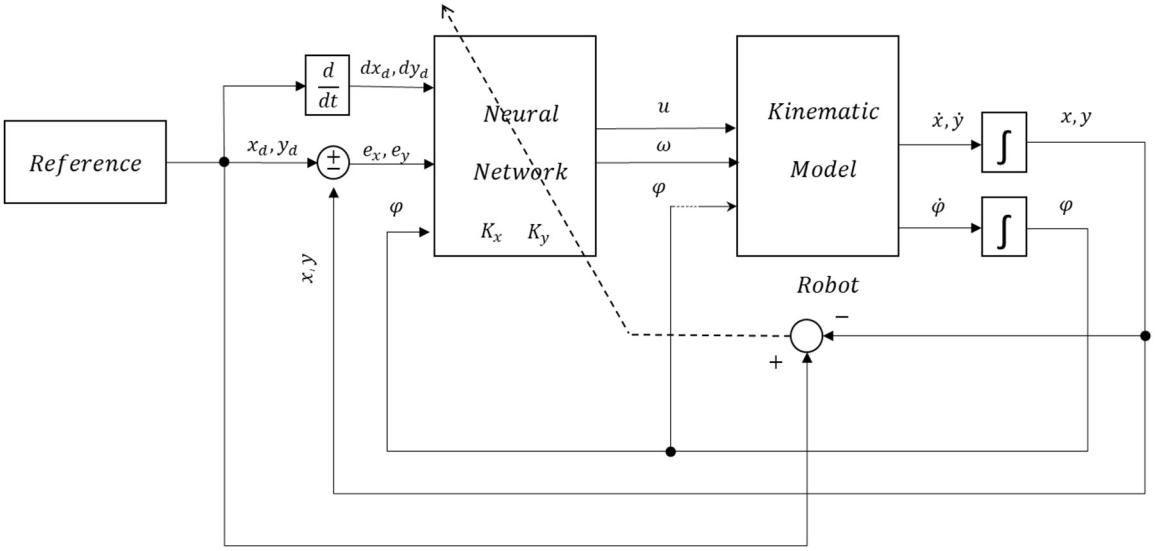


Figura 2.5 Modelo de Referencia.

La estructura propuesta de la red neuronal y el controlador basado en redes neuronales utilizado para conducir el sistema a la referencia se muestran en el diagrama de bloques de la **Figura 2.5**.

A partir de las leyes de control del controlador cinemático se realiza el diseño de la red neuronal.

Desarrollando la ley de control de la ecuación (2.21) se tiene:

$$u = \cos\varphi \ (\dot{x_d} + k_x \, e_x) + \sin\varphi \ \left(\dot{y_d} + k_y \, e_y\right) \tag{2.29}$$

$$\omega = -\frac{1}{a}\sin\varphi \ (\dot{x_d} + k_x \, e_x) + \frac{1}{a}\cos\varphi \ \left(\dot{y_d} + k_y \, e_y\right) \tag{2.30}$$

La distancia del punto desplazado será de: $a = 0.1m$

Con las ecuaciones (2.29) y (2.30) se diseña la red neuronal siguiendo el esquema de un perceptrón multicapa. La **Figura 2.6**. muestra el diseño de la red neuronal diseñada a partir del modelo del controlador cinemático de un robot móvil.

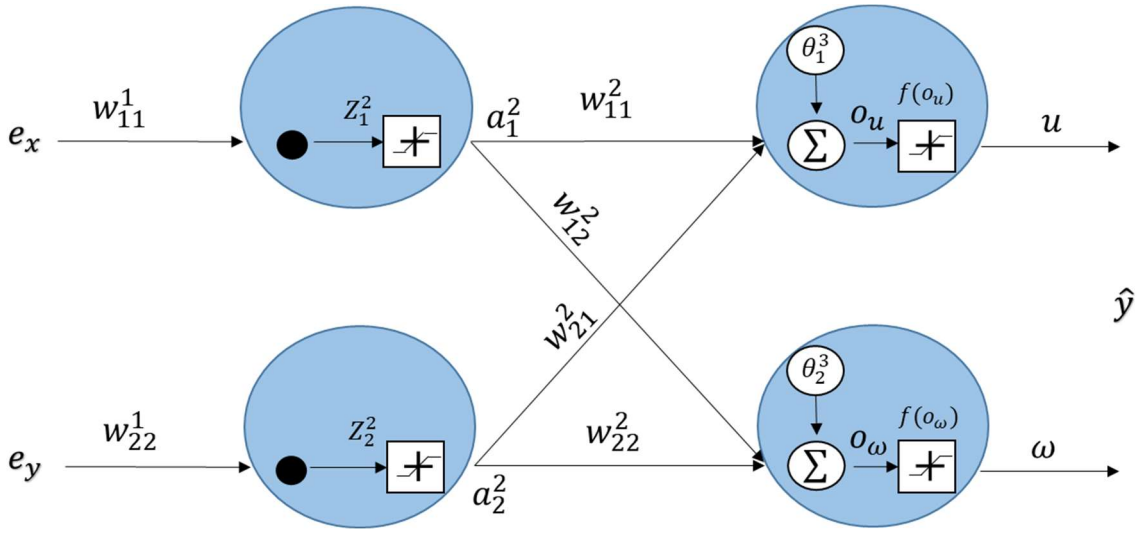


Figura 2.6 Red Neuronal Diseñada a partir del Modelo del Controlador Cinemático Robot Móvil.

Los pesos sinápticos en la capa de entrada están dados por:

$$w_{11}^1 = k_x \tag{2.31}$$

$$w_{22}^1 = k_y \tag{2.32}$$

La capa de entrada está conectada 1 a 1 con la primera capa oculta dado que los pesos w_{12}^1 y w_{21}^1son cero. También, no se consideran umbrales en la primera capa oculta. Estas consideraciones se toman a partir del diseño de la red neuronal a partir del modelo del controlador cinemático del robot móvil mostrado en la **Figura 2.6**.

Las ganancias k_x y k_y corresponden los pesos sinápticos de la red neuronal que se ajustarán para que el sistema alcance una rápida convergencia a la referencia deseada disminuyendo el error de posición. Además, la función de activación utilizada es la tangente hiperbólica descrita por la ecuación (2.6). Esta función sirve como mecanismo de saturación de las señales de control u y ω, garantizando que sus valores no superen ciertos límites impuestos por los actuadores. Puede tomar valores tanto positivos como negativos, impidiendo el envío de señales de control que superen los niveles de energía permitidos.

La activación de las neuronas en la capa de entrada se halla con la ecuación (2.1) y viene dada por:

$$a_1^1 = e_x \tag{2.33}$$

$$a_2^1 = e_y \tag{2.34}$$

En la capa oculta se tienen las siguientes relaciones:

$$Z_1^2 = e_x\, k_x \tag{2.35}$$

$$Z_2^2 = e_y\, k_y \tag{2.36}$$

La activación de las neuronas en la capa oculta se halla con la ecuación (2.2) y viene dada por:

$$a_1^2 = f(Z_1^2) = \tanh(e_x\, k_x) \tag{2.37}$$

$$a_2^2 = f(Z_2^2) = \tanh\left(e_y\, k_y\right) \tag{2.38}$$

Los pesos sinápticos en la capa oculta vienen dados por:

$$w_{11}^2 = \cos\varphi \tag{2.39}$$

$$w_{12}^2 = -\frac{1}{a}\sin\varphi \tag{2.40}$$

$$w_{21}^2 = \sin\varphi \tag{2.41}$$

$$w_{22}^2 = \frac{1}{a}\cos\varphi \tag{2.42}$$

Los umbrales en la capa de salida vienen dados por:

$$\theta_1^3 = \dot{x_d}\cos\varphi + \dot{y_d}\sin\varphi \tag{2.43}$$

$$\theta_2^3 = -\frac{1}{a}sin\varphi\ \dot{x_d} + \frac{1}{a}cos\varphi\ \dot{y_d} \tag{2.44}$$

La activación de las neuronas de la capa de salida se halla con la ecuación (2.3) viene dada por:

$$u = a_1^3 = \tanh(o_u) \tag{2.45}$$

$$\omega = a_2^3 = \tanh(o_\omega) \tag{2.46}$$

Siendo

$$o_u = cos\varphi\ a_1^2 + sin\varphi\ a_2^2 + \theta_1^3 \tag{2.47}$$

$$o_\omega = -\frac{1}{a} sin\varphi\ a_1^2 + \frac{1}{a} cos\varphi\ a_2^2 + \theta_2^3 \tag{2.48}$$

El objetivo del controlador diseñado es que se asegure de que la salida de la planta siga la referencia del modelo, con el mínimo valor de error posible para ambos ejes x, y. Las ganancias k_x y k_y del controlador se irán ajustando por medio del algoritmo de backpropagation hasta que el error entre la trayectoria actual y la trayectoria deseada sea aproximadamente cero.

El error total del seguimiento de la trayectoria está descrito por:

$$e(n) = \frac{1}{2}\left(e^2_x + e^2_y\right) \tag{2.49}$$

De acuerdo con la ecuación (2.9) y teniendo en cuenta que el número de muestras escogidos será de $N = 1$, dado que el seguimiento de trayectoria se realizará en tiempo real y mientras más grande sea el número de muestras, más tiempo le tomará al robot en alcanzar la referencia, entonces se tiene que:

$$E = e(n) \tag{2.50}$$

Las ganancias k_x y k_y , corresponderán a los pesos sinápticos que se desean adaptar para minimizar la ecuación (2.49). Por lo que el aprendizaje se plantea como un problema de minimización de la forma:

$$Min_{k_x,k_y} E \tag{2.51}$$

A partir de la ecuación (2.11), las leyes de aprendizaje se definen como:

$$k_x(n) = k_x(n-1) - \alpha \frac{\partial E(n)}{\partial k_x} \tag{2.52}$$

$$k_y(n) = k_y(n-1) - \alpha \frac{\partial E(n)}{\partial k_y} \tag{2.53}$$

Para resolver la ecuación (2.52) y (2.53) es necesario resolver las siguientes ecuaciones:

$$\frac{\partial E}{\partial k_x} = \frac{\partial E}{\partial x} + \frac{\partial E}{\partial y} \tag{2.54}$$

$$\frac{\partial E}{\partial k_y} = \frac{\partial E}{\partial x} + \frac{\partial E}{\partial y} \tag{2.55}$$

Mediante la regla de la cadena para las ecuaciones (2.54) y (2.55) se tiene que:

$$\begin{aligned}\frac{\partial E}{\partial k_x} = {} & \frac{\partial E}{\partial x}\left[\left(\frac{\partial x}{\partial u}\frac{\partial u}{\partial o_u}\frac{\partial o_u}{\partial a_1^2} + \frac{\partial x}{\partial \omega}\frac{\partial \omega}{\partial o_\omega}\frac{\partial o_\omega}{\partial a_1^2}\right)\frac{\partial a_1^2}{\partial Z_1^2}\frac{\partial Z_1^2}{\partial k_x}\right] \\ & + \frac{\partial E}{\partial y}\left[\left(\frac{\partial y}{\partial u}\frac{\partial u}{\partial o_u}\frac{\partial o_u}{\partial a_1^2} + \frac{\partial y}{\partial \omega}\frac{\partial \omega}{\partial o_\omega}\frac{\partial o_\omega}{\partial a_1^2}\right)\frac{\partial a_1^2}{\partial Z_1^2}\frac{\partial Z_1^2}{\partial k_x}\right]\end{aligned} \tag{2.56}$$

$$\begin{aligned}\frac{\partial E}{\partial k_y} = {} & \frac{\partial E}{\partial x}\left[\left(\frac{\partial x}{\partial u}\frac{\partial u}{\partial o_u}\frac{\partial o_u}{\partial a_2^2} + \frac{\partial x}{\partial \omega}\frac{\partial \omega}{\partial o_\omega}\frac{\partial o_\omega}{\partial a_2^2}\right)\frac{\partial a_2^2}{\partial Z_2^2}\frac{\partial Z_2^2}{\partial k_y}\right] \\ & + \frac{\partial E}{\partial y}\left[\left(\frac{\partial y}{\partial u}\frac{\partial u}{\partial o_u}\frac{\partial o_u}{\partial a_2^2} + \frac{\partial y}{\partial \omega}\frac{\partial \omega}{\partial o_\omega}\frac{\partial o_\omega}{\partial a_2^2}\right)\frac{\partial a_2^2}{\partial Z_2^2}\frac{\partial Z_2^2}{\partial k_y}\right]\end{aligned} \tag{2.57}$$

A continuación, se hallan las derivadas parciales necesarias para resolver las ecuaciones (2.56) y (2.57).

Despejando e_x de la ecuación (2.29) y reemplazando la ecuación (2.19) se tiene que:

$$x = x_d - \frac{u}{k_x \cos\varphi} + \frac{\sin\varphi\left(\dot{y_d} + k_y\, e_y\right)}{k_x \cos\varphi} + \frac{\dot{x_d}}{k_x} \tag{2.58}$$

Despejando e_y de la ecuación (2.29) y reemplazando la ecuación (2.19) se tiene que:

$$y = y_d - \frac{u}{k_y \sin\varphi} + \frac{\cos\varphi\left(\dot{x_d} + k_x\, e_x\right)}{k_y \sin\varphi} + \frac{\dot{y_d}}{k_y} \tag{2.59}$$

Despejando e_x de la ecuación (2.30) y reemplazando la ecuación (2.19) se tiene que:

$$x = x_d - \frac{\cos\varphi \left(\dot{y_d} + k_y\, e_y\right)}{k_x \sin\varphi} + \frac{\omega\, a}{k_x \sin\varphi} + \frac{\dot{x_d}}{k_x} \tag{2.60}$$

Despejando e_y de la ecuación (2.30) y reemplazando la ecuación (2.19) se tiene que:

$$y = y_d - \frac{\sin\varphi\, (\dot{x_d} + k_x\, e_x)}{k_y \cos\varphi} - \frac{\omega\, a}{k_y \cos\varphi} + \frac{\dot{y_d}}{k_y} \tag{2.61}$$

Obteniendo las siguientes derivadas parciales:

$$\frac{\partial x}{\partial u} = -\frac{1}{k_x \cos\varphi} \tag{2.62}$$

$$\frac{\partial y}{\partial u} = -\frac{1}{k_y \sin\varphi} \tag{2.63}$$

$$\frac{\partial x}{\partial \omega} = \frac{a}{k_x \sin\varphi} \tag{2.64}$$

$$\frac{\partial y}{\partial \omega} = -\frac{a}{k_y \cos\varphi} \tag{2.65}$$

De la ecuación (2.49) se obtienen las siguientes derivadas parciales:

$$\frac{\partial E}{\partial x} = -e_x \tag{2.66}$$

$$\frac{\partial E}{\partial y} = -e_y \tag{2.67}$$

De las ecuaciones (2.45) (2.46) y reemplazando la ecuación (2.7) se obtienen las siguientes derivadas parciales:

$$\frac{\partial u}{\partial o_u} = f'(o_u) = 1 - \tanh^2(o_u) \tag{2.68}$$

$$\frac{\partial \omega}{\partial o_\omega} = f'(o_\omega) = 1 - \tanh^2(o_\omega) \tag{2.69}$$

De las ecuaciones (2.47) (2.48), se obtienen las siguientes derivadas parciales:

$$\frac{\partial o_u}{\partial a_1^2} = \cos\varphi \tag{2.70}$$

$$\frac{\partial o_\omega}{\partial a_1^2} = -\frac{1}{a}\sin\varphi \tag{2.71}$$

$$\frac{\partial o_u}{\partial a_2^2} = \sin\varphi \tag{2.72}$$

$$\frac{\partial o_\omega}{\partial a_2^2} = \frac{1}{a}\cos\varphi \tag{2.73}$$

Considerando la derivada de la función de activación descrita por la ecuación (2.7), se obtienen las siguientes derivadas parciales:

$$\frac{\partial a_1^2}{\partial Z_1^2} = 1 - \tanh^2(e_x k_x) \tag{2.74}$$

$$\frac{\partial a_2^2}{\partial Z_2^2} = 1 - \tanh^2(e_y k_y) \tag{2.75}$$

De las ecuaciones (2.35) y (2.36) se obtienen las siguientes derivadas parciales:

$$\frac{\partial Z_1^2}{\partial k_x} = e_x \tag{2.76}$$

$$\frac{\partial Z_2^2}{\partial k_y} = e_y \tag{2.77}$$

2.7 Desarrollo de las Trayectorias de Referencia

El control de seguimiento de trayectoria del robot humanoide se aplicará en tres trayectorias diferentes.

Las trayectorias probadas son: Circular (2.78), Lemniscata (2.79), Cuadrada (2.80).

El punto inicial para las trayectorias (2.78), (2.79), (2.80) será igual a $(x, y) = (0.05\ , 0.05)\ [m]$.

$$\begin{cases} x_{ref}(t) = \cos\left(\dfrac{t}{120}\right) \\ y_{ref}(t) = \sin\left(\dfrac{t}{120}\right) \end{cases} \tag{2.78}$$

$$\begin{cases} x_{ref}(t) = \sin\left(\frac{t}{140}\right) \\ y_{ref}(t) = \sin\left(\frac{t}{70}\right) \end{cases} \tag{2.79}$$

$$\begin{cases} x_{ref}(t) = 0.5 - \frac{t}{200} \ \forall\ t\ \epsilon\ [0{,}200];\ -0.5\ \forall\ t\ \epsilon\ [200{,}400]; \\ \frac{1}{200} * (t - 400) - 0.5\ \ \forall\ t\ \epsilon\ [400{,}600]; 0.5\ \forall\ t\ \epsilon\ [600{,}800]; \\ y_{ref}(t) = 0.5\ \forall\ t\ \epsilon [0{,}200]; 0.5 - \frac{1}{200} * (t - 200)\ \forall\ t\ \epsilon\ [200{,}400]; \\ -0.5\ \forall\ t\ \epsilon [400{,}600]; \frac{1}{200} * (t - 600) - 0.5\ \forall\ t\ \epsilon\ [600{,}800]; \end{cases} \tag{2.80}$$

Con estas trayectorias se realizarán las comparaciones gráficas y numéricas para comprobar el desempeño de los controladores diseñados en esta tarea de control.

2.8 Configuración Entornos de Simulación para el Robot NAO

Para que el robot NAO interactúe con VREP y Simulink, es necesario utilizar el lenguaje de programación Python y programar las instrucciones correspondientes de envío y recepción de datos. El entorno de ejecución de Python actuará como intermediario entre ambos programas, permitiendo la transferencia de datos entre ellos.

2.8.1 Entorno de Ejecución de PYTHON

En el entorno de ejecución de Python, es indispensable llevar a cabo la comunicación en paralelo para garantizar la ejecución simultánea de los datos. Para ello, se emplea el comando *threading*, el cual permite procesar y ejecutar los datos de manera concurrente. Asimismo, es fundamental definir la estructura de cada articulación del robot NAO para lograr su conexión con las articulaciones correspondientes en VREP. Esto se puede lograr mediante la definición de la función *JointControl*. Adicionalmente, se requiere la librería NAOqi de Choregraphe en el entorno, para poder ejecutar funciones como caminar y mantener la postura del robot NAO.

2.8.2 VREP- PYTHON

Para permitir la comunicación entre VREP y Python, es necesario configurar la API remota en un cuboide. Este cuboide debe ubicarse en una parte no visible del entorno de simulación y contener la programación de la API en su estructura tal como se observa en la **Figura 2.7**.

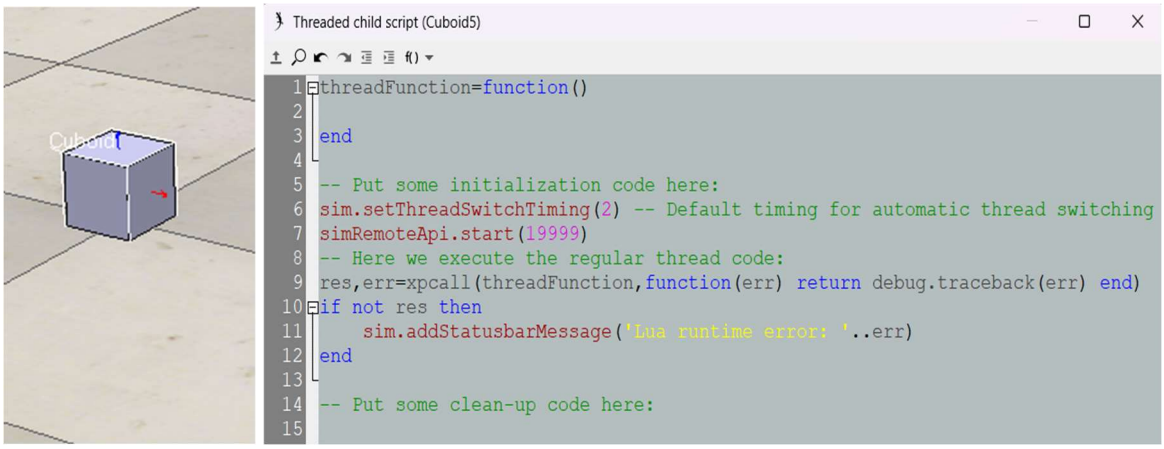


Figura 2.7 Configuración del Cuboide en VREP.

La interfaz de navegación de VREP con el robot NAO se muestra en la **Figura 2.8**.

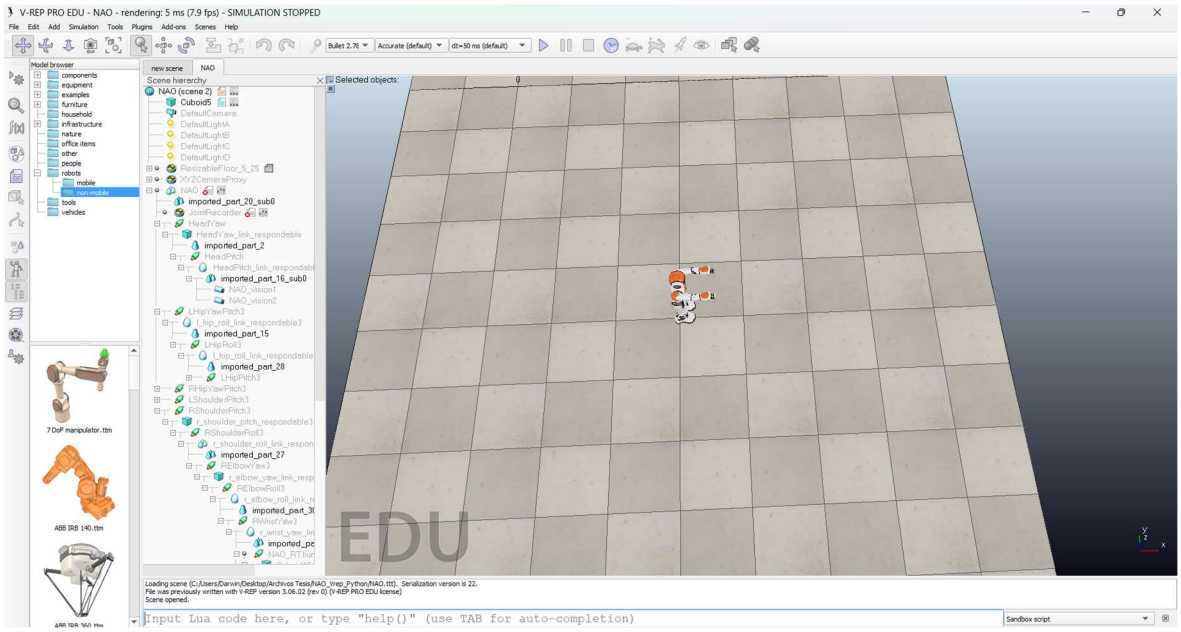


Figura 2.8 Entorno de Simulación VREP.

En el entorno de ejecución de Python es necesario agregar la línea de código mostrada en la **Figura 2.9** para iniciar la comunicación necesaria.

```
clientID=vrep.simxStart('127.0.0.2',19999,True,True,5000,5) #Connection VREP
```

Figura 2.9 Código para Conexión a VREP.

El controlador requiere conocer la posición y orientación del robot, los cuales se obtienen a partir de los sensores del mismo. Para obtener estos valores, se utilizan los comandos mostrados en la **Figura 2.10** y **Figura 2.11**. El primer comando proporciona la posición en metros, mientras que el segundo proporciona la orientación en radianes.

```
NAO_pos = vrep.simxGetObjectPosition(clientID, NAO_Handle[1], -1, vrep.simx_opmode_streaming)[1]
```

Figura 2.10 Comando para la obtención de la Posición del Robot.

```
NAO_ang = vrep.simxGetObjectOrientation(clientID, NAO_Handle[1], -1, vrep.simx_opmode_streaming)[1]
```

Figura 2.11 Comando para la obtención de la Orientación del Robot.

2.8.3 Librería NAOqi

Para permitir que el robot NAO pueda caminar y adoptar posturas, se requiere definir el objeto ALProxy que adquirirá las características del módulo correspondiente. Este objeto, representado en la **Figura 2.12** y **Figura 2.13**, contiene las funciones de movimiento y postura necesarias para el funcionamiento del robot. Asimismo, es fundamental especificar la dirección IP y el puerto de conexión de Choregraphe del robot.

```
moveProxy = ALProxy("ALMotion",NaoIP, NaoPort)
```

Figura 2.12 ALProxy de Movimiento NAOqi.

```
postureProxy = ALProxy("ALRobotPosture", NaoIP, NaoPort)
```

Figura 2.13 ALProxy de Postura NAOqi.

Una vez definidos los objetos ALProxy, es necesario que los datos de velocidad lineal y angular, obtenidos desde Simulink, permitan el movimiento del robot NAO a partir del comando de la **Figura 2.14**. Las velocidades estarán en m/s y rad/s.

```
moveProxy.move(v,0,w)
```

Figura 2.14 Comandos para Movimiento del Robot NAO.

2.8.4 SIMULINK- PYTHON

En Simulink, se llevará a cabo la configuración de los bloques UDP receive y UDP send para realizar el envío y la recepción de datos, respectivamente, mediante el protocolo de comunicación UDP. Los datos se empaquetarán en un array, siendo los valores de velocidad lineal y angular los datos que se enviarán, mientras que los datos que se recibirán corresponderán a las posiciones del robot, que previamente fueron capturados en VREP.

En la **Figura 2.15** se observa el esquema de envío y recepción de datos en Simunlink, es necesario que los valores se discreticen en un tiempo de muestreo de 0.02 s, dado que

este es el tiempo de respuesta del robot NAO. De igual forma, se deberá especificar un número de puerto y una dirección IP.

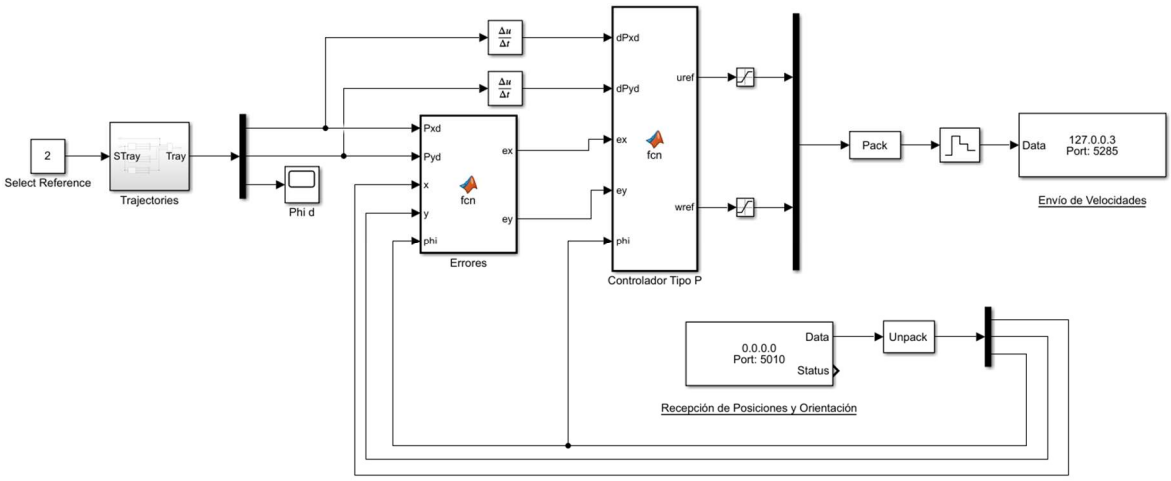


Figura 2.15 Esquema de Envío y Recepción de datos en Simulink.

En el entorno de ejecución de Python se debe declarar los comandos para la comunicación UDP. Estos comandos se encuentran en la librería socket. De igual forma se deberán especificar los puertos y dirección IP declarados en Simulink.

En la **Figura 2.16** y **Figura 2.17** se muestran los comandos necesarios para la recepción y envío de datos entre Simulink y el entorno de ejecución de Python.

```
sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
sock.bind(('127.0.0.3',5285))
aux='>'+('d'*2)
data, addr = sock.recvfrom(1024)
mensaje=struct.unpack(aux, data)
sock.shutdown(0)
sock.close
print 'Mensaje recibido: ' , mensaje
v=mensaje[0]
w=mensaje[1]
```

Figura 2.16 Recepción de datos desde Simulink al Entorno de Ejecución de Python.

```
data=[pos_x,pos_y,phi]
sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
sock.bind(('127.0.0.3',5011))
aux='>'+('d'*len(data))
mensajebody=struct.pack(aux,*data)
sock.sendto (mensajebody, ('127.0.0.3',5010))
print 'Mensaje enviado: ',data
```

Figura 2.17 Envío de datos desde el Entorno de Ejecución de Python a Simulink.

2.8.5 Diagrama de Flujo del Programa Principal

En la **Figura 2.18** se muestra el diagrama de flujo de la estructura del programa principal utilizado en la elaboración del proyecto.

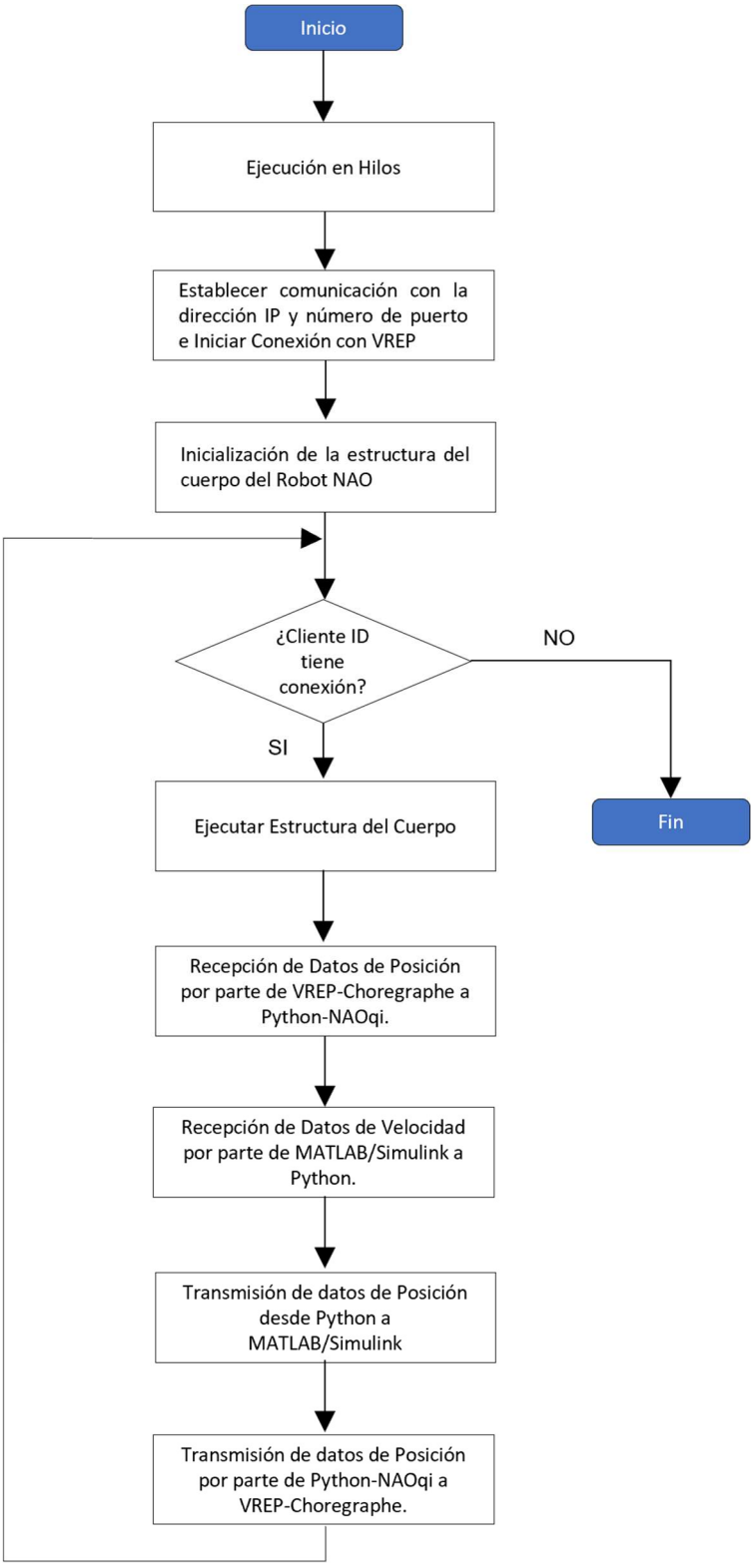


Figura 2.18 Estructura del Programa Principal.

2.9 Desarrollo Interfaz Gráfica

En el diseño de la interfaz gráfica se utiliza la norma ISO 9241-151, que establece varias directrices para la estructura de la interfaz de usuario. Entre estas directrices destacan: la necesidad de que la interfaz esté centrada, que sea sencilla y tenga una presentación coherente, que los comandos sean breves y que se eviten los colores muy vivos para evitar fatiga visual [48].

A través de la interfaz gráfica, el operador podrá controlar los parámetros de los controladores, seleccionar la trayectoria de referencia y observar el seguimiento de trayectoria del robot NAO. La interfaz cuenta con 4 pestañas. La primera, mostrada en la **Figura 2.19**, corresponde a la pantalla principal que proporciona información sobre el proyecto. La senda pestaña, mostrada en la **Figura 2.20**, muestra las instrucciones para iniciar la simulación del programa. En la tercera pestaña, mostrada en la **Figura 2.21**, se pueden controlar los parámetros del controlador seleccionado y elegir la trayectoria de referencia. Además, en esta pantalla se puede ejecutar la simulación y observar el seguimiento de trayectoria del robot. Por último, en la cuarta pantalla, mostrada en la **Figura 2.22**, se podrán visualizar las variaciones de las ganancias del controlador y las señales de control y error del seguimiento de trayectoria del robot NAO.

Figura 2.19 Pantalla Principal.

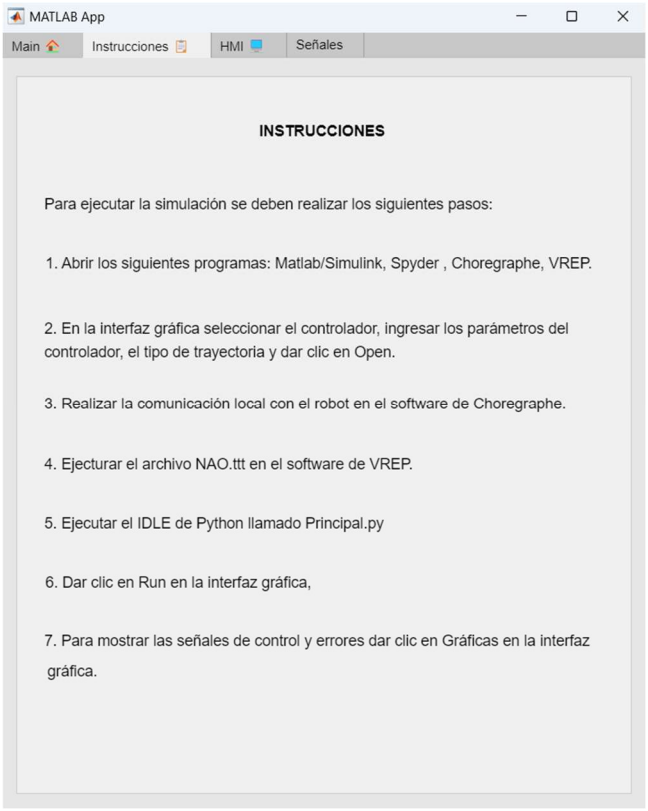


Figura 2.20 Pantallas de Instrucciones.

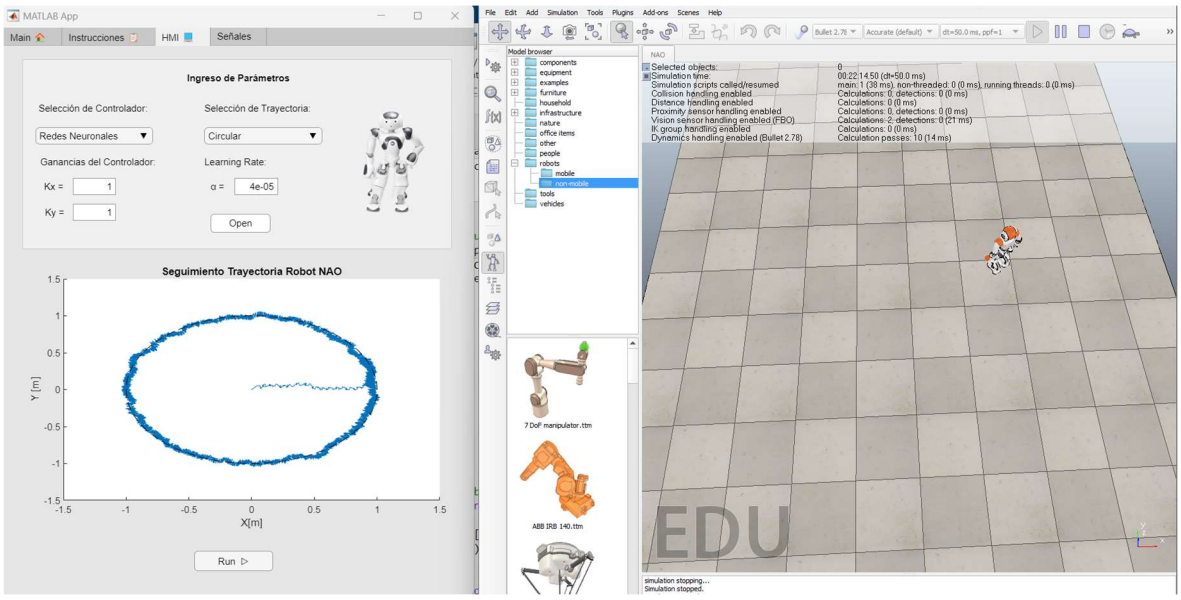


Figura 2.21 Pantalla de Control del Seguimiento de Trayectoria.

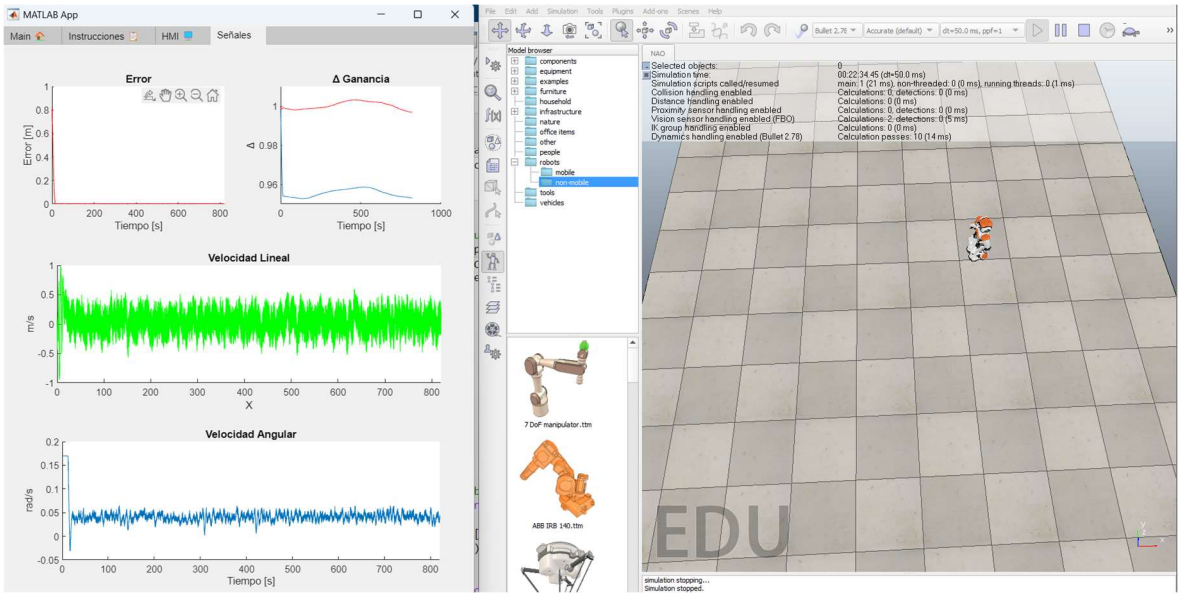


Figura 2.22 Pantalla de Señales de Control y Error.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En este apartado se presentan las pruebas de simulación del controlador basado en redes neuronales, comprobando que el robot humanoide NAO realiza el seguimiento de la trayectoria de referencia. A continuación, se evalúa su desempeño comparándolo con un controlador cinemático convencional mediante índices de desempeño utilizados en sistemas de control. Además, se incluyen las conclusiones y recomendaciones.

3.1 Resultados

Los parámetros iniciales para el controlador cinemático convencional (2.21) y para el controlador basado en redes neuronales (2.29), (2.30) se presentan en la **Tabla 3.1** y **Tabla 3.2** respectivamente.

Tabla 3.1 Parámetros Iniciales para el Controlador Cinemático Convencional.

	Control Tipo P		
Valor Parámetro	***Trayectorias***		
	Circular	*Lemniscata*	*Cuadrada*
k_p	1		

Tabla 3.2 Parámetros Iniciales para el Controlador Basado en Redes Neuronales.

	Control Basado en Redes Neuronales		
Valores Parámetros	***Trayectorias***		
	Circular	*Lemniscata*	*Cuadrada*
k_x	1		
k_y	1		
α	0.00004	0.0002	0.00004

La **Figura 3.1** muestra el seguimiento del robot humanoide NAO para una trayectoria circular con radio de 1m, mientras que la **Figura 3.2** presenta la variación de las ganancias del controlador cinemático basado en redes neuronales. Con el controlador cinemático convencional tipo P, las señales de control de la velocidad lineal y angular presentan oscilaciones bruscas y de mayor magnitud que pueden afectar a la vida útil de los actuadores del robot. Estas oscilaciones se reducen cuando el robot alcanza la trayectoria deseada. Con el controlador cinemático basado en redes neuronales, estas oscilaciones

se reducen y suavizan debido al autoajuste de las ganancias de la planta, de modo que la trayectoria actual converge rápidamente a la trayectoria deseada.

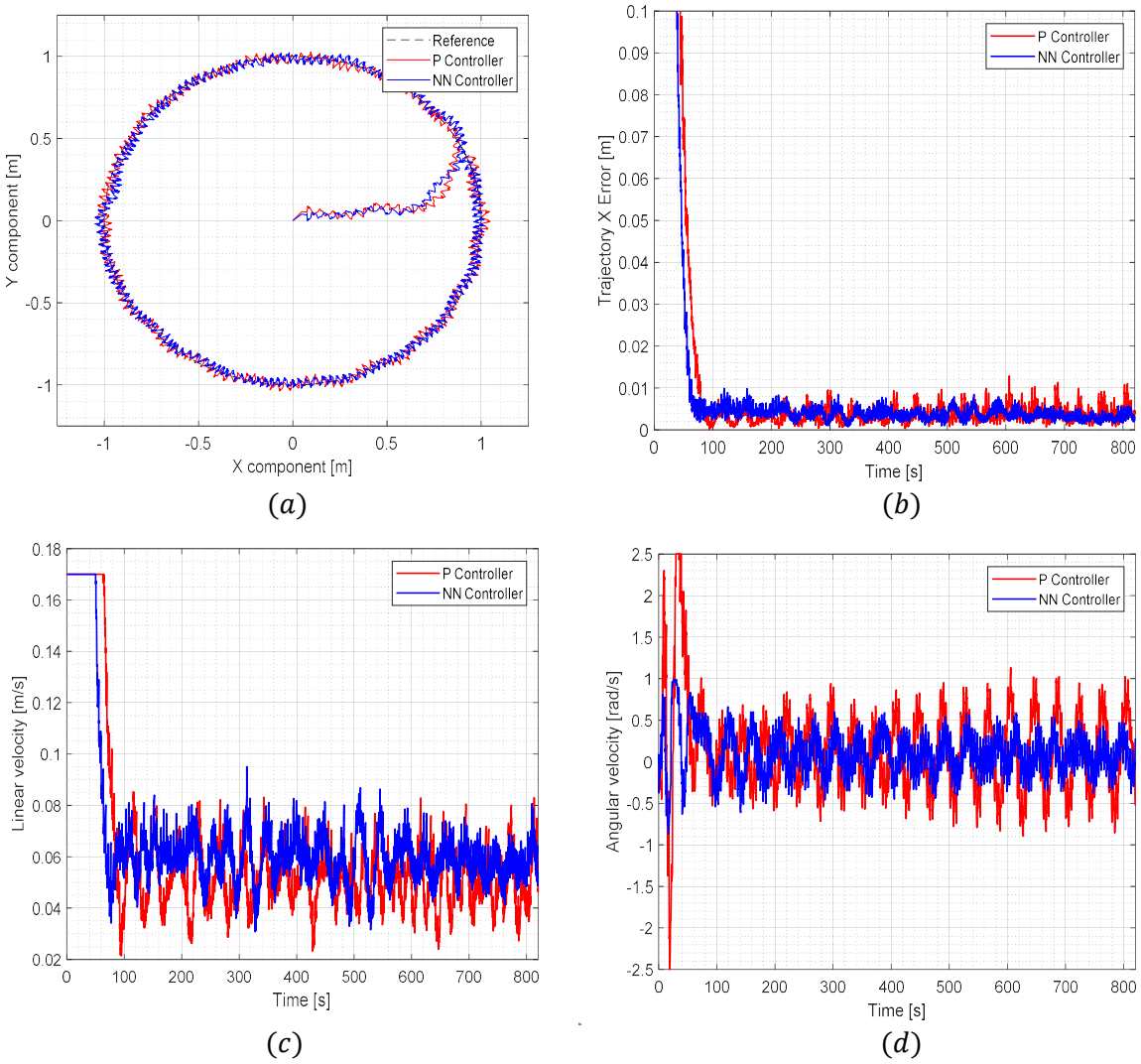


Figura 3.1 (a) Trayectoria circular realizada por el robot, (b) error de posición del robot, (c) velocidad linear y (d) velocidad angular.

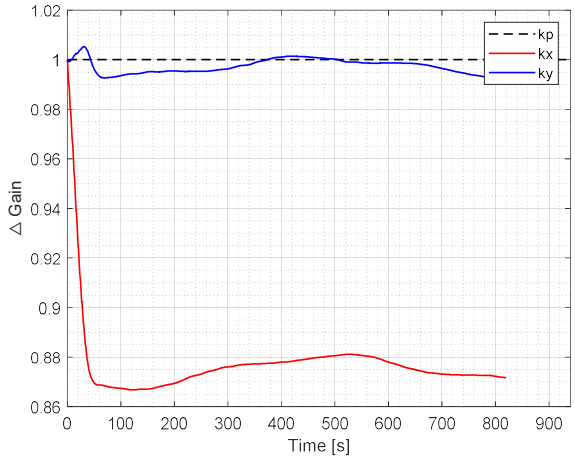


Figura 3.2 Variación Ganancias Controlador basado en Redes Neuronales Círculo.

La **Figura 3.3** muestra el seguimiento del robot humanoide para una trayectoria lemniscata de radio de 1m, mientras que la **Figura 3.4** presenta la variación de las ganancias del controlador cinemático basado en redes neuronales. El robot humanoide alcanza la trayectoria rápidamente. Sin embargo, en el controlador convencional tipo P se siguen presentando oscilaciones que pueden afectar en la vida útil de los actuadores de la planta. Con el controlador cinemático basado en redes neuronales no se alcanzan los valores de saturación de la velocidad lineal. Y el error de seguimiento de la trayectoria es mucho menor al del controlador convencional tipo P.

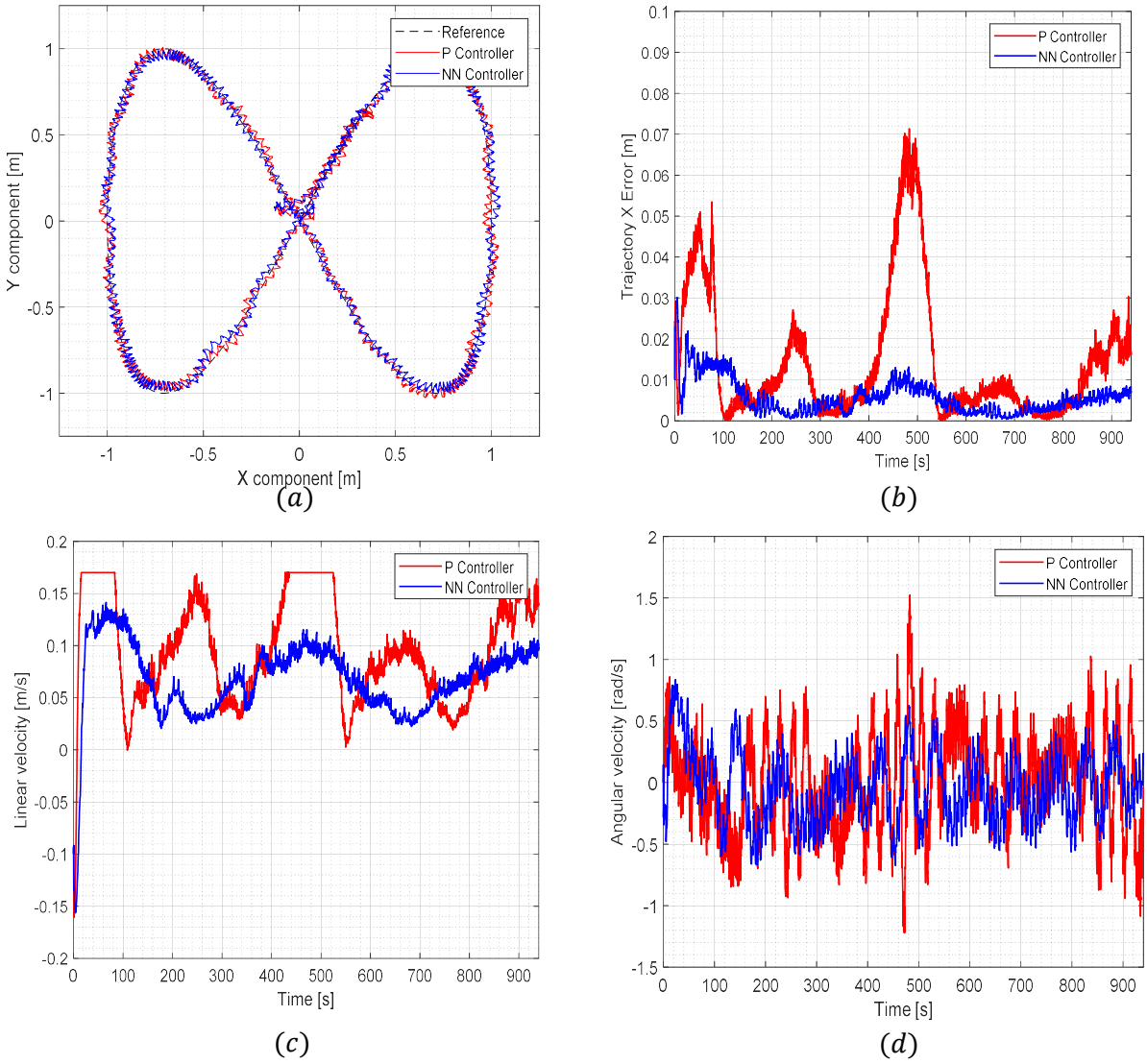


Figura 3.3 (a) Trayectoria lemniscata realizada por el robot, (b) error de posición del robot, (c) velocidad linear y (d) velocidad angular.

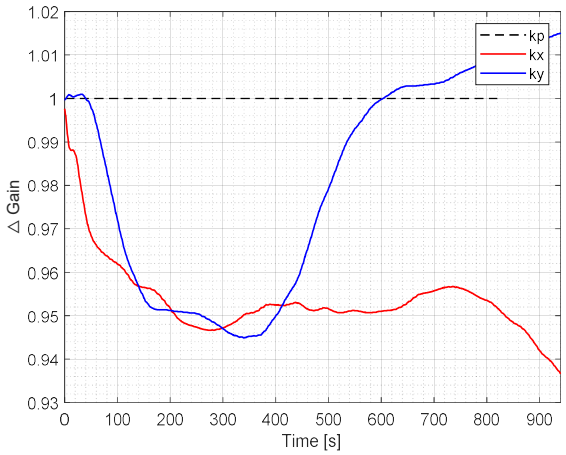


Figura 3.4 Variación Ganancias Controlador basado en Redes Neuronales Lemniscata.

La **Figura 3.5** muestra el seguimiento del robot humanoide para una trayectoria cuadrada con lado de 1m, mientras que la **Figura 3.6** presenta la variación de las ganancias del controlador cinemático basado en redes neuronales. A partir del autoajuste de las ganancias del controlador cinemático basado en redes neuronales, se reducen las oscilaciones bruscas y de gran magnitud en comparación con el controlador convencional tipo P. En la trayectoria cuadrada, se observa que el error aumenta en las esquinas debido a los cambios de orientación del robot, pero se observa que este error se corrige más rápido con el controlador cinemático basado en redes neuronales.

Figura 3.1, **Figura 3.3** y **Figura 3.5** muestran los resultados comparativos de la trayectoria circular, lemniscata y cuadrada, respectivamente, donde se muestran los resultados de un controlador convencional tipo P y un controlador cinemático basado en redes neuronales se muestran, así como el error de posición en cada trayectoria. Además, se muestra la forma de las señales de velocidad lineal y angular.

Figura 3.1 (b), **Figura 3.3** (b) y **Figura 3.5** (b) muestran el seguimiento del robot humanoide. Los parámetros del controlador convencional tipo P y del controlador cinemático basado en redes neuronales tienen los mismos valores y los resultados muestran que el controlador cinemático basado en redes neuronales tiene mejor desempeño con respecto al controlador convencional tipo P, obteniendo un pequeño error de distancia con respecto a las referencias. El error tiene un valor no tan grande considerando que no se consideró el modelo cinemático o dinámico del robot humanoide para el diseño de los controladores. Con la red neuronal hay un menor error debido a su capacidad de aproximar sistemas no lineales.

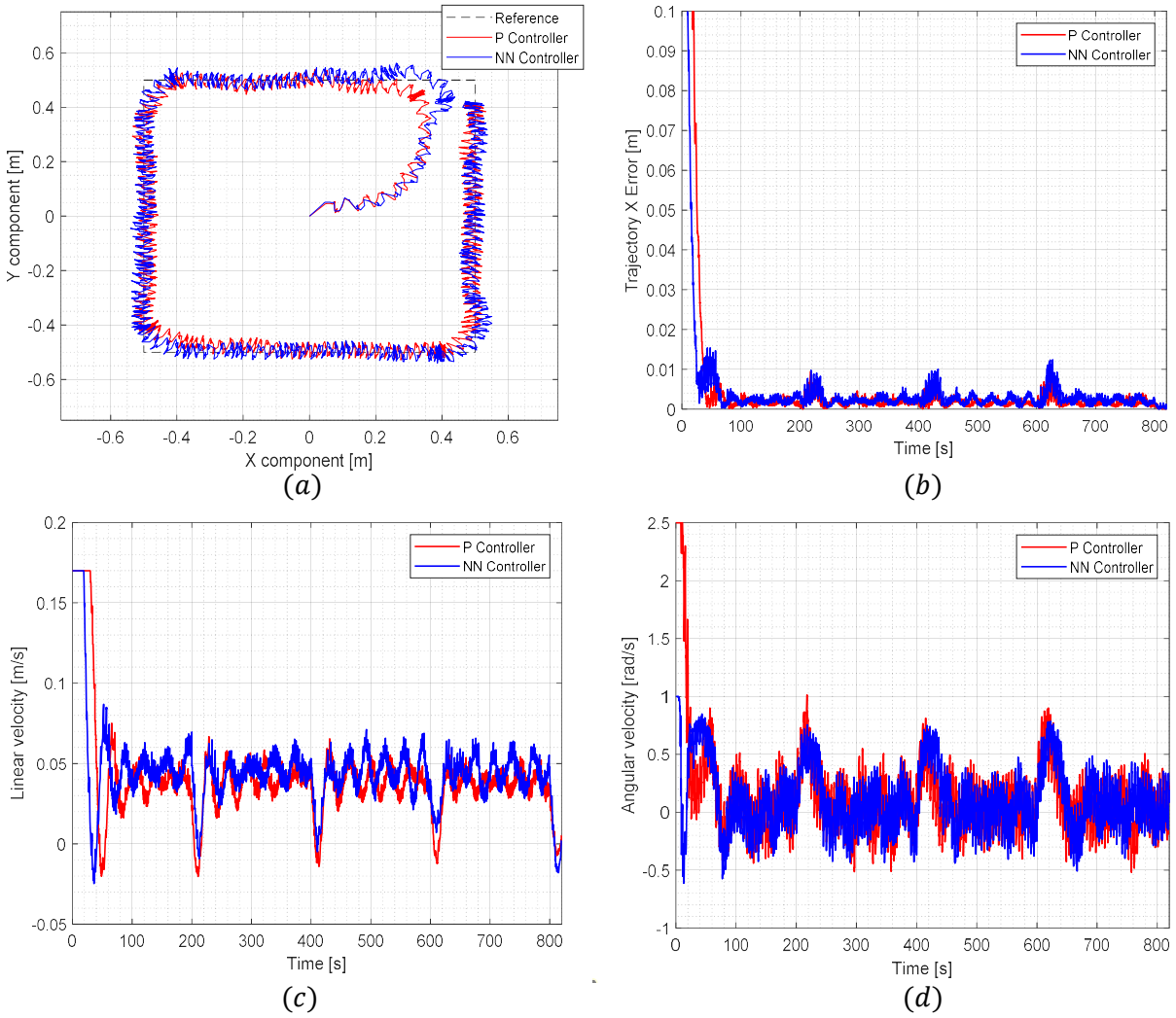


Figura 3.5 (a) Trayectoria cuadrada realizada por el robot, (b) error de posición del robot, (c) velocidad linear y (d) velocidad angular.

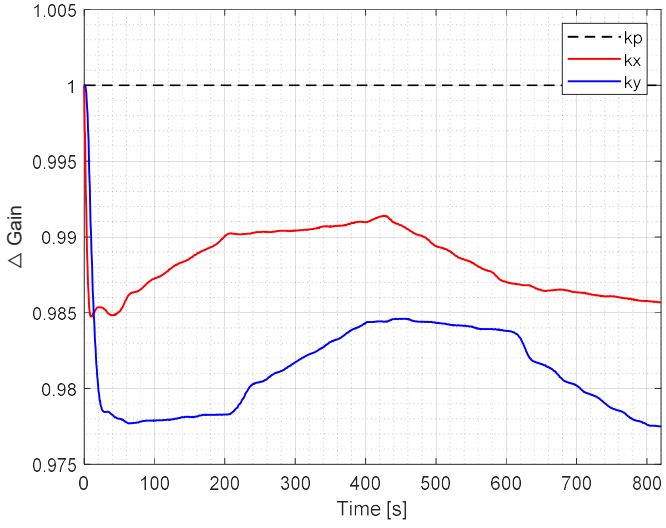


Figura 3.6 Variación Ganancias Controlador basado en Redes Neuronales Cuadrada.

La velocidad lineal en todas las trayectorias tiene aproximadamente un valor de $0.06m/s$ (**Figura 3.1** (c), **Figura 3.3** (c) y **Figura 3.5** (c)), la velocidad obtenida por el controlador neuronal tiene un buen desempeño y rápida convergencia, mostrando en términos de eficiencia que el controlador cinemático basado en redes neuronales tiene una respuesta más suave y mejor en términos cuantitativos si analizamos los valores IAE, ISE, ISU en la **Figura 3.7**, **Figura 3.8** y **Figura 3.9**. Los parámetros de los controladores tienen los mismos valores. Sin embargo, los valores en el controlador cinemático basado en redes neuronales varían hasta alcanzar el valor óptimo con el que se minimice el error de la trayectoria.

Para el caso de la velocidad angular (**Figura 3.1** (d), **Figura 3.3** (d) y **Figura 3.5** (d)) los resultados muestran que se reducen las oscilaciones y se obtiene señales más suaves y dentro de los rangos esperados. Por lo que se observa que este comportamiento permite que el sistema converja rápidamente al valor de referencia. La reducción de las oscilaciones para esta señal de control se disminuye considerablemente, lo que es una ventaja destacada del uso de este controlado basado en redes neuronales.

La variación de las ganancias del controlador se puede observar en la **Figura 3.2**, **Figura 3.4** y **Figura 3.6** y es claro que con estos valores se reduce el error. Además, la señal de velocidad se encuentra saturada debido al uso de la función de activación tangente hiperbólica, por lo que la señal obtenida con este controlador reduce el número de oscilaciones en las acciones de control, lo que permite que la vida útil de los actuadores no se vea tan afectada.

Los resultados en la **Figura 3.7**, **Figura 3.8** y **Figura 3.9**, muestran una mejora cuantitativa en términos de rendimiento del controlador basado en redes neuronales con respecto al controlador convencional tipo P. En el caso de los índices de desempeño IAE, ISE, e ISU de la señal de control de velocidad lineal, el controlador cinemático basado en redes neuronales presenta una mejora mayor al 10% sobre el controlador convencional tipo P. Pero en el caso del índice de desempeño ISU de la señal de control de velocidad angular, el controlador basado en redes neuronales presenta una mejora considerable. Esto debido a que se reduce el número de oscilaciones. Debido a estas razones, el controlador propuesto tiene un mejor comportamiento en el sistema debido a una rápida convergencia al valor de referencia sin la necesidad del modelo cinemático o dinámico del robot.

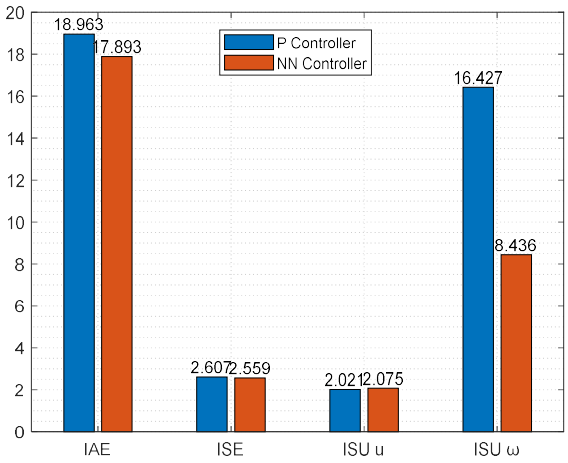


Figura 3.7 Índices de Desempeño Trayectoria Circular.

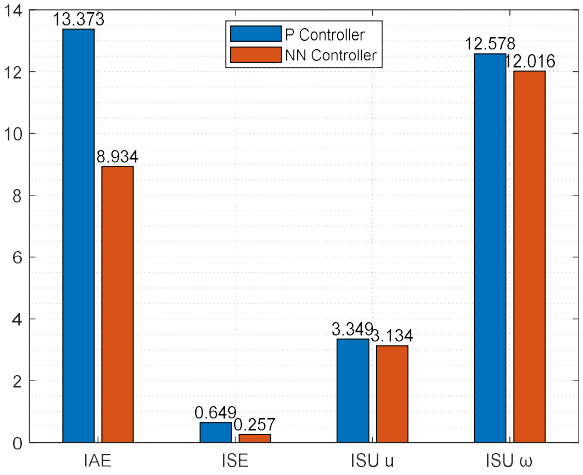


Figura 3.8 Índices de Desempeño Trayectoria Lemniscata.

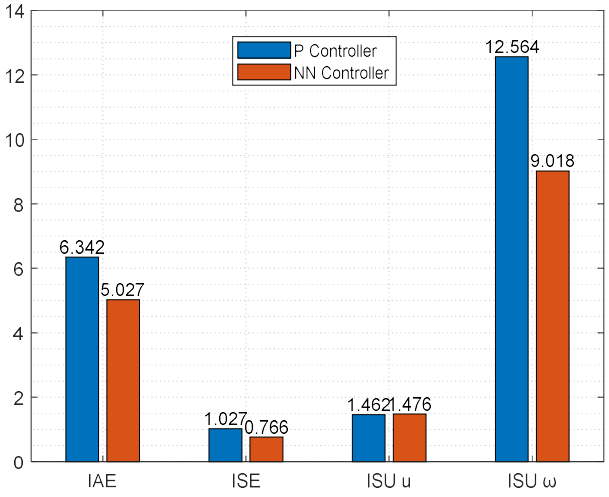


Figura 3.9 Índices de Desempeño Trayectoria Cuadrada.

3.2 Conclusiones

En este trabajo se ha presentado el diseño de un controlador basado en redes neuronales. A partir del modelo del controlador cinemático de un robot móvil, se diseña la red neuronal, siendo las ganancias del controlador los parámetros que se ajustarán con el algoritmo de backpropagation.

El controlador propuesto, en términos cualitativos, controla de mejor manera que un controlador convencional tipo P, que es un controlador utilizado habitualmente para el seguimiento de trayectorias. Así, es evidente que el controlador cinemático basado en redes neuronales es mucho mejor que el controlador convencional tipo P, porque el error de trayectoria es menor y las señales de control de la velocidad lineal y angular no son tan oscilatorias, lo que conserva la vida útil de los actuadores.

En términos cuantitativos, el método propuesto tiene muy buenos resultados en comparación con el controlador convencional tipo P. En el seguimiento de trayectoria del robot humanoide, con el método propuesto, el robot converge más rápido a la referencia que a diferencia que el controlador convencional tipo P. Además, que las acciones de control reducen considerablemente el número de oscilaciones que se refleja en el índice de desempeño ISU, lo que representa que la energía empleada por el controlador será menor. El sistema de control es estable y no necesita del modelo matemático de la cinemático o dinámico de la planta.

3.3 Recomendaciones

Es necesario que el tiempo de muestreo de 20 ms del robot NAO, esté en los bloques de Simulink, para realizar el correcto envío y recepción de datos. Este tiempo también debe ser utilizado como tiempo de ejecución de las instrucciones del robot. Si se tiene mayor o menor tiempo de muestreo puede haber pérdida o saturación de datos respectivamente.

Tomar en cuenta los diferentes parámetros establecidos por el fabricante del robot humanoide, para que las señales de control no afecten el comportamiento del robot.

En trabajos futuros de investigación, se podría utilizar el controlador cinemático basado en redes neuronales con el uso de sensores de proximidad para que el robot humanoide NAO evite obstáculos.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] Pedro Isasi Viñuela and Inés M. Galván León, *Redes de Neuronas Artificiales Un Enfoque Práctico*. Madrid: PEARSON EDUCACIÓN, S.A., 2004.

[2] S. Sarker, L. Jamal, S. F. Ahmed, and N. Irtisam, "Robotics and artificial intelligence in healthcare during COVID-19 pandemic: A systematic review," *Robotics and Autonomous Systems*, vol. 146. 2021. doi: 10.1016/j.robot.2021.103902.

[3] I. tecnológico virtual IAeñ™ and ™ T. B. of A., "Inteligencia Artificial en Medicina," *IA eñ ™*. 2020. doi: 10.21428/39829d0b.844809d3.

[4] C. Ramió Matas, "Inteligencia artificial, robótica y modelos de Administración pública," *Revista del CLAD Reforma y Democracia*, no. 72, 2018.

[5] S. G. Tzafestas, *Introduction to Mobile Robot Control*. 2013. doi: 10.1016/C2013-0-01365-5.

[6] B. E. Byambasuren, D. Kim, M. Oyun-Erdene, C. Bold, and J. Yura, "Inspection robot based mobile sensing and power line tracking for smart grid," *Sensors (Switzerland)*, vol. 16, no. 2, 2016, doi: 10.3390/s16020250.

[7] J. M. Bengochea-Guevara, J. Conesa-Muñoz, D. Andújar, and A. Ribeiro, "Merge fuzzy visual servoing and GPS-based planning to obtain a proper navigation behavior for a small crop-inspection robot," *Sensors (Switzerland)*, vol. 16, no. 3, 2016, doi: 10.3390/s16030276.

[8] T. Klamt *et al.*, "Flexible Disaster Response of Tomorrow: Final Presentation and Evaluation of the CENTAURO System," *IEEE Robot Autom Mag*, vol. 26, no. 4, 2019, doi: 10.1109/MRA.2019.2941248.

[9] L. Xu, J. Du, B. Song, and M. Cao, "A combined backstepping and fractional-order PID controller to trajectory tracking of mobile robots," *Systems Science and Control Engineering*, vol. 10, no. 1, 2022, doi: 10.1080/21642583.2022.2047125.

[10] C. Kahraman, M. Deveci, E. Boltürk, and S. Türk, "Fuzzy controlled humanoid robots: A literature review," *Rob Auton Syst*, vol. 134, 2020, doi: 10.1016/j.robot.2020.103643.

[11] S. Parsianmehr, S. A. A. Moosavian, and A. Fakharian, "An experimental system identification modeling and robust control for NAO humanoid robot," in *4th RSI International Conference on Robotics and Mechatronics, ICRoM 2016*, 2017. doi: 10.1109/ICRoM.2016.7886793.

[12] Z. Pang, B. Zhang, J. Yu, Z. Sun, and L. Gong, "Design and analysis of a Chinese Medicine based Humanoid Robotic Arm Massage System," *Applied Sciences (Switzerland)*, vol. 9, no. 20, 2019, doi: 10.3390/app9204294.

[13] J. J. Alcaraz-Jiménez, D. Herrero-Pérez, and H. Martínez-Barberá, "Robust feedback control of ZMP-based gait for the humanoid robot Nao," *International Journal of Robotics Research*, vol. 32, no. 9–10, 2013, doi: 10.1177/0278364913487566.

[14] N. Hassan and A. Saleem, “Neural Network-Based Adaptive Controller for Trajectory Tracking of Wheeled Mobile Robots,” *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3146970.

[15] S. Shamsuddin *et al.*, “Humanoid robot NAO: Review of control and motion exploration,” in *Proceedings - 2011 IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2011*, 2011. doi: 10.1109/ICCSCE.2011.6190579.

[16] M. Sato *et al.*, “Rehabilitation care with Pepper humanoid robot: A qualitative case study of older patients with schizophrenia and/or dementia in Japan,” *Enferm Clin*, vol. 30, 2020, doi: 10.1016/j.enfcli.2019.09.021.

[17] M. Fujita, “AIBO: Toward the era of digital creatures,” *International Journal of Robotics Research*, vol. 20, no. 10, 2001, doi: 10.1177/02783640122068092.

[18] T. Kanda, T. Hirano, D. Eaton, and H. Ishiguro, “Interactive robots as social partners and peer tutors for children : A field trial,” *Hum Comput Interact*, vol. 19, no. 1–2, 2004, doi: 10.1207/s15327051hci1901&2_4.

[19] S. Thrun *et al.*, “MINERVA: a second-generation museum tour-guide robot,” *Proc IEEE Int Conf Robot Autom*, vol. 3, 1999, doi: 10.1109/robot.1999.770401.

[20] J. Y. C. Chen, “UAV-guided navigation for ground robot tele-operation in a military reconnaissance environment,” *Ergonomics*, vol. 53, no. 8, 2010, doi: 10.1080/00140139.2010.500404.

[21] T. Shibata and K. Wada, “Robot therapy: A new approach for mental healthcare of the elderly - A mini-review,” *Gerontology*, vol. 57, no. 4. 2011. doi: 10.1159/000319015.

[22] S. M. Orozco-Soto and J. M. Ibarra-Zannatha, “Control con rechazo activo de perturbaciones para el equilibrio de robots humanoides,” *Research in Computing Science*, vol. 135, no. 1, 2017, doi: 10.13053/rcs-135-1-11.

[23] SoftBank Robotics, “NAO6 the versatile humanoid robot,” *Press Kit*. pp. 1–7.

[24] SoftBank Robotics, “NAO6 Guía del Usuario,” *Instrucciones de uso de NAO*, pp. 1–10, 2019.

[25] SoftBank Robotics, “Nao6 Datasheet,” *H25600*. pp. 1–6, 2018.

[26] E. Pot, J. Monceaux, R. Gelin, B. Maisonnier, and A. Robotics, “Choregraphe: A graphical tool for humanoid robot programming,” in *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, 2009. doi: 10.1109/ROMAN.2009.5326209.

[27] A. Ramkumar, U. Akhil Krishna, M. S. Madhan, and K. K. Prajit, “Control of Nao robot arm using Myo Armband,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 9 Special Issue 2, 2019, doi: 10.35940/ijitee.I1087.0789S219.

[28] B.-C. Eusebio and A.-B. Ana Yaveni, “Control visual para la formación de robots móviles tipo uniciclo bajo el esquema líder-seguidor,” *Ingeniería, Investigación y Tecnología*, vol. 15, no. 4, 2014, doi: 10.1016/s1405-7743(14)70657-2.

[29] L. Morales, M. Herrera, O. Camacho, P. Leica, and J. Aguilar, “LAMDA Control Approaches Applied to Trajectory Tracking for Mobile Robots,” *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3062202.

[30] K. Malu and J. Majumdar, “Kinematics, Localization and Control of Differential Drive Mobile Robot,” *Global Journal of Researches in Engineering*, vol. 14, no. 1, pp. 1–8, 2014, [Online]. Available: http://www.engineeringresearch.org/index.php/GJRE/article/view/1233

[31] F. G. Rossomando, C. Soria, and R. Carelli, “Sliding mode neuro adaptive control in trajectory tracking for mobile robots,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 74, no. 3–4, 2014, doi: 10.1007/s10846-013-9843-5.

[32] S. Liu, H. Zhang, S. X. Yang, and J. Yu, “Dynamic control of a mobile robot using an adaptive neurodynamics and sliding mode strategy,” in *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, 2004. doi: 10.1109/wcica.2004.1343669.

[33] L. Gracia and J. Tornero, “Kinematic control of wheeled mobile robots,” *Latin American Applied Research*, vol. 38, no. 1, 2008.

[34] F. G. Rossomando, C. Soria, and R. Carelli, “Control de Robots Móviles con Incertidumbres Dinámicas usando Redes de Base Radial,” *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 7, no. 4, 2010, doi: 10.1016/s1697-7912(10)70057-1.

[35] Z. Wu and H. Du, “Artificial Intelligence in Agricultural Picking Robot Displacement Trajectory Tracking Control Algorithm,” *Wirel Commun Mob Comput*, vol. 2022, 2022, doi: 10.1155/2022/3105909.

[36] L. Chen, P. Chen, and Z. Lin, “Artificial Intelligence in Education: A Review,” *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2988510.

[37] E. Zalama, M. Paul, and J. R. Perán, “Neural Network for the Behavioral Navigation of a Mobile Robot,” *IFAC Proceedings Volumes*, vol. 31, no. 3, 1998, doi: 10.1016/s1474-6670(17)44067-5.

[38] G. N. Marichal, J. Toledo, L. Acosta, E. J. González, and G. Coll, “A neuro-fuzzy method applied to the motors of a stereovision system,” *Eng Appl Artif Intell*, vol. 20, no. 7, 2007, doi: 10.1016/j.engappai.2006.12.010.

[39] C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” *Electronic Markets*, vol. 31, no. 3, 2021, doi: 10.1007/s12525-021-00475-2.

[40] E. Mendez *et al.*, “ANN based MRAC-PID controller implementation for a furuta pendulum system stabilization,” *Advances in Science, Technology and Engineering Systems*, vol. 5, no. 3, 2020, doi: 10.25046/aj050342.

[41] Mathias Thor, “Introduction to V-REP.” Coppelia Robotics AG, pp. 1–51.

[42] E. Rohmer, S. P. N. Singh, and M. Freese, “V-REP: A versatile and scalable robot simulation framework,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326. doi: 10.1109/IROS.2013.6696520.

[43] A. Kaviyarasu, A. Saravanakumar, and M. Logavenkatesh, “Software in loop simulation based waypoint navigation for fixed wing uav,” *Def Sci J*, vol. 71, no. 4, 2021, doi: 10.14429/DSJ.71.16164.

[44] A. R. Valdez Alvarado, “Machine Learning para Todos.,” *Reseaechgate*, no. January 2019, 2019.

[45] J. Jepkoech, D. M. Mugo, B. K. Kenduiywo, and E. C. Too, “The Effect of Adaptive Learning Rate on the Accuracy of Neural Networks,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, 2021, doi: 10.14569/IJACSA.2021.0120885.

[46] F. N. Martins, M. Sarcinelli-Filho, and R. Carelli, “A Velocity-Based Dynamic Model and Its Properties for Differential Drive Mobile Robots,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 85, no. 2, 2017, doi: 10.1007/s10846-016-0381-9.

[47] J. A. Valencia V, A. O. Montoya, L. Hernando Rios, and M. Sc Profesor Titular, “Modelo cinemático de un robot móvil tipo diferencial y navegación a partir de la estimación odométrica.,” *Scientia et Technica, ISSN 0122-1701, Vol. 1, Nº. 41, 2009, págs. 191-196*, vol. 1, no. 41, 2009.

[48] G. Martínez, “Ergonomía e interfases de interacción humano-computadora,” México, D.F, 2007.