UNIVERSIDAD SANTO TOMÁS

DISEÑO DE UN CONTROLADOR APLICADO A UN PÉNDULO INVERTIDO UTILIZANDO ESTRATEGIAS BASADAS EN APRENDIZAJE DE MÁQUINA

Realizado por

Julián Rincón Martínez

Gustavo Alonso Pineda Gonzalez

Proyecto de grado presentado como requisito para optar al título de
INGENIERO ELECTRÓNICO

Grupo de Investigación GED (Grupo de Estudio y Desarrollo en Robótica)
Facultad de Ingeniería Electrónica
División de Ingenierías

Septiembre de 2021

DISEÑO DE UN CONTROLADOR APLICADO A UN PÉNDULO INVERTIDO UTILIZANDO ESTRATEGIAS BASADAS EN APRENDIZAJE DE MÁQUINA

Realizado por

Julián Rincón Martínez

Gustavo Alonso Pineda Gonzalez

Proyecto de grado presentado como requisito para optar al título de
INGENIERO ELECTRÓNICO

Dirigido por

José Guillermo Guarnizo Marín

Co-Dirigido por

Edgar Camilo Camacho Poveda

Grupo de Investigación GED (Grupo de Estudio y Desarrollo en Robótica)
Facultad de Ingeniería Electrónica
División de Ingenierías

Septiembre de 2021

AGRADECIMIENTOS

Primero que todo, el agradecimiento es para las personas que han estado siempre apoyándonos, nuestros padres, que siempre nos motivan para alcanzar todos los objetivos que nos proponemos y que incluso en los momentos más complejos siguen creyendo en nosotros, sin ese apoyo emocional no hubiese sido posible culminar con éxito este proyecto de grado.

También a los docentes que siempre han estado dispuestos a ayudarnos a ser mejores profesionales y mejores personas día a día, que con paciencia y esmero siempre tratan de impartir de la mejor manera sus conocimientos para con nosotros.

Por último a nuestros compañeros y colegas que también han estado a nuestro lado brindándonos apoyo.

Índice general

Índice de figuras

Resumen

Este documento presenta el desarrollo de controlador aplicado a un péndulo invertido simple, utilizando estrategias basadas en aprendizaje de máquina. Para el desarrollo de este proyecto, se hace uso de una plataforma simulada en el software Simulink, la cual realiza una respectiva caracterización del sistema y una representación de la planta a partir de diagrama de bloques implementado en este Software. En este documento se muestran las ecuaciones no lineales de la planta para que, a partir del comportamiento de esta, se diseñe un control basado en redes neuronales que sea capaz de estabilizar la posición angular del péndulo invertido alrededor de un punto de trabajo específico. El desarrollo del proyecto se divide en 4 fases importantes.

La primera fase consiste en conocer las variables de entrada y salida de la planta que se usarán para el diseño de la red neuronal. La variable de entrada hace referencia al voltaje inyectado a la planta, mientras que las variables de salida hacen referencia a los ángulos del péndulo y del brazo giratorio.También se muestran las ecuaciones que describen el comportamiento del sistema de péndulo invertido. En la segunda fase se hace una revisión en el estado del arte, esto con el fin de observar metodologías implementadas en trabajos anteriores, tomando como partida algunos métodos de aprendizaje de máquina que puedan ser usados para el control de posición angular en un péndulo. La tercera fase consiste en la obtención de los datos de la planta que serán los usados en el diseño de las redes neuronales como conjunto de datos de entrenamiento y pruebas. A partir de los datos obtenidos en la tercera fase, en la cuarta fase se implementa la red neuronal con su respectivo entrenamiento y será la encargada de estabilizar el péndulo.

Se presentan los resultados experimentales que se llevaron a cabo sobre las redes neuronales implementadas, teniendo en cuenta diferentes pruebas realizadas, haciendo cambios en parámetros como los retardos, los ciclos, la frecuencia de muestreo o los datos de entrada. También

se hace un análisis de estas pruebas y de las gráficas obtenidas del sistema a partir de parámetro de error, con el fin observar el comportamiento del péndulo una vez la red neuronal ha sido entrenada.

Introducción

En la actualidad el estudio de controles inteligentes se ha venido enfocando en el desarrollo industrial para el cumplimiento de tareas de una forma más efectiva, esto, entre otras cosas, intentando reducir los tiempos de desempeño y de producción a nivel industrial, científico y comercial, tanto así que estos se han desplazado a la vida cotidiana para ayudar a mejorar la condición de vida de las personas [1]. Para lograr entender como funciona un brazo robótico, el posicionamiento de un satélite o la navegación de robots bípedos hay que realizar un arduo estudio tanto matemático - físico como electrónico - mecánico. Estos son algunos de los usos que puede llegar a solucionar a partir del análisis en péndulos invertidos.

La exploración de un control usando técnicas de aprendizaje de máquina para la estabilización de un péndulo invertido simple trata de encontrar un método para poder estabilizar y controlar éste sistema, basándose en algoritmos ya existentes en machine learning o sistemas bioinspirados basados en aprendizaje por redes neuronales o, en su defecto, diseñar un nuevo algoritmo basado en estas mismas técnicas puesto que los métodos mas usados para esta clase de problemas hacen uso de controladores tradicionales los cuales representan una complejidad mayor.

Dicho esto, este proyecto inicia formulando la problemática observada y a partir de esta busca hacer un estudio en técnicas de aprendizaje de máquina para estabilizar el péndulo invertido simple sobre una plataforma simulada, esto teniendo de base la investigación mostrada en el estado del arte. Seguidamente se seleccionan la métrica que se usará para validar el sistema implementado, todo esto con la revisión del estado del arte. Luego, se realiza la toma de datos que servirán de base de entrenamiento para la red neuronal que, una vez entrenada, se implementará en el sistema del péndulo invertido, buscando mostrar cómo es su funcionamiento con respecto a un sistema implementado con un control clásico como el Control LQG (Linear Quadratic Gaussian Control - Control Lineal Cuadrático Gaussiano).

Capítulo 1

Planteamiento del Problema

Los sistemas no lineales representan grandes retos para la ingeniería electrónica, debido a la complejidad presentada en la implementación de técnicas que sean capaces de controlar y estabilizar dichos sistemas. El péndulo invertido simple muestra un sistema altamente no lineal, cuya linealización representa una dificultad matemática alta, a pesar de esto, el péndulo invertido ha sido objeto de estudio e investigación a lo largo de los años, en donde se han implementado una gran cantidad de técnicas de control para poder interactuar con este. La mayoría de estudios realizados sobre un sistema de péndulo invertido han usado técnicas convencionales de control, como se puede apreciar en la revisión del estado del arte. Estas técnicas implican un complejo desarrollo matemático y en muchos casos la implementación física ha traído problemas mecánicos que hacen más difícil el desarrollo de controladores.

Los avances ingenieriles actuales han llevado a la generación de problemas y aplicaciones relacionados a su control que son de carácter no lineal, esto se puede evidenciar en fenómenos no lineales como equilibrio, sistemas de comando de vuelo o manipuladores robóticos [2], estos problemas se ven reflejados matemáticamente cuando se busca hacer una linealización de estos sistemas, lo que ha llevado a la migración hacia técnicas de control que integre nuevas formas para lograr una estabilización de manera más sencilla [3].

El sistema de péndulo invertido muestra un punto de vista muy amplio que ha sido de vital ayuda para la implementación y estudio de controladores, esto debido a que se ilustran dificultades asociadas al control del mundo real y esto resulta importante al momento de realizar análisis sobre sistemas que deben permanecer estabilizados sobre un punto específico. El sistema de un péndulo invertido presenta dificultades por su no linealidad [4].

Ante el hecho de conocer la dificultad matemática de trabajar con sistemas no lineales se busca estudiar técnicas de control por aprendizaje de máquina que ayuden a facilitar la interacción con estos sistemas no lineales, por lo que se plantea la siguiente pregunta que busca representar la idea general del proyecto. ¿Cómo integrar técnicas de aprendizaje de máquina en un sistema de control aplicada a sistemas no lineales como el péndulo invertido, para lograr su estabilización?

Capítulo 2

Antecedentes

Actualmente existen múltiple prácticas e investigaciones basadas en sistemas no lineales como el péndulo invertido, es importante, de antemano, revisar algunas de estas investigaciones que se enlistan en los antecedentes, con el fin de dar una idea un poco más ajena a la realidad y los avances que se están dando actualmente, para lograr abordar con mayor prevención un estudio en la exploración de un control inteligente que logre la estabilización y control del péndulo. En dichos antecedentes se puede observar sistemas implementados desde controles de un péndulo invertido por controlador neuro-difuso para la educación en control inteligente [5], hasta la implementación en carro o riel con un sistema de péndulo invertido en tiempo real capaz de mantener dicho péndulo en posición vertical mientras que la base (carro o riel) se desplaza de manera horizontal [6]. Para la realización de estos trabajos se tuvieron en cuenta varias técnicas de control, como el análisis de controles lineales y no lineales, control implementado por Matlab, control predictivo, control con redes neuronales hasta controles basados en aprendizaje OpenFlow.

En [5] lo que se busca es la estabilización de un péndulo invertido basado en técnicas de control Neuro-difuso con objetivo de poder tener una educación sobre controles inteligentes, el algoritmo de dicho control fue implementado sobre una tarjeta de desarrollo DSP 2812 y con esto aseguraban que el control se realizaba en tiempo real, la conclusión de este trabajo muestra que a pesar que el algoritmo neuro-difuso mantiene un margen de estabilidad adecuado, se tuvo que implementar un sistema mucho más robusto basado en esta misma técnica al inicial debido a que en un principio cuando se implementa dicho sistema el péndulo no se lograba estabilizar del todo.

En [6] se implementa un péndulo invertido sobre una plataforma robótica, la base del diseño de dicho controlador se enfoca en uno por espacio de estados donde se implementa la matemática para su linealización.

En [7] se busca la estabilidad de un péndulo en posición vertical sobre una cama de aire, este sistema representa un modelo de sistema inestable no lineal, y su objetivo principal es estabilizar dicho péndulo que al más pequeño movimiento o empuje sobre éste se caerá, para lograr contrarrestar dicha inestabilidad, en caso tal de que el péndulo sea empujado, es provocando una fuerza motora en el planeador (cama de aire) o motor que se encargará de equilibrar el péndulo. El control del sistema fue diseñado de manera analítica y seguidamente se implementó digitalmente sobre un microcontrolador, que se encargaba de activar un motor con cierta fuerza moderada sobre el péndulo y de esta manera mantenerlo en equilibrio. Los resultados de esta práctica permitieron controlar el sistema hasta por 8 minutos, un tiempo que no garantiza una gran estabilidad cuando se trataba de perturbaciones lo suficientemente fuertes impidiendo al motor del sistema estabilizar el péndulo.

En [8], [9] y [10] el trabajo realizado es similar entre ellos, en donde [8] consiste en la solución de balanceo y control de un péndulo, que será empujado o balanceado desde una posición 'cero' verticalmente hacia abajo hasta que este se encuentre en posición vertical hacia arriba, desde allí aplicar un control específico para garantizar su estabilidad en ese punto final. Este trabajo mezcla dos tipos de control, los cuales se dividen en un control capaz de garantizar una fuerza de balanceo, que permita al péndulo llegar a su posición vertical y desde este punto, se hace el cálculo analítico para garantizar la estabilidad del péndulo en esta región. En [9] tratan un problema muy similar donde se busca el balanceo y control de un péndulo invertido, en este trabajo se hizo una comparativa entre la teoría de control difusa, con respecto a estrategias de oscilación y control basado en energía, este concluye que en sistemas o aplicaciones no lineales en tiempo real la mejor alternativa se enfoca en la teoría de control difusa. En [10] se busca la implementación de un carro o riel que es capaz de estabilizar un péndulo invertido de manera autónoma, este sistema que se basa en una implementación total del sistema por parte de los autores muestra un control digital implementado sobre una tarjeta de desarrollo situada en el carro, dicho controlador se basa en la técnica de retroalimentación de estados usando Root Locus.

En [11] se muestra un control para péndulo invertido implementado en el software Matlab por la técnica de variables de estados y usando también un controlador LQR (Linear Quadratic Regulator - Regulador Lineal Cuadrático).

En [12] se muestra un control híbrido para la estabilización de un sistema de péndulo invertido capaz de garantizar dicha estabilidad siendo capaz de adaptarse a múltiples condiciones de perturbación externa, este tipo de controlador híbrido presenta una ventaja frente a controladores convencionales según los autores del trabajo respecto a la adaptación del sistema. Los controles que se implementan son un control difuso y un control LQR (Linear Quadratic Regulator - Regulador Lineal Cuadrático), con estos controladores se logra recopilar la información más importante del sistema y de los controladores, con dicha información se pone en marcha el entrenamiento del sistema adaptativo de inferencia neurodifusa. Con la combinación de los dos controladores entrenando, como una red neuronal permitiendo al sistema obtener mejor rendimiento a la hora de estabilizar el péndulo.

En [13] refleja la implementación de un sistema de aprendizaje profundo que permite trabajar sobre una planta de péndulo invertido rotatorio no lineal. Dicho sistema está implementado en un entorno simulado y un entorno físico basado en la red OpenFlow como también del protocolo MQTT, que se usa en la conexión que une o conecta ambos entornos por medio de Ethernet. En esta práctica se usa un PID clásico para la implementación del aprendizaje por refuerzo e imitación, facilitando el proceso de aprendizaje.

En [14], [15], [16] y [17] se muestra la implementación de controladores basados en habilidades mediante el uso de red neuronal difusa para control inteligente jerárquico, control mediante el uso de redes neuronales y mapas cognitivos difusos, todos aplicados para control inteligente y aunque estos artículos no se trabajan sobre un péndulo invertido, muestran técnicas que pueden llegar a ser útil para la elaboración de este proyecto, usando el entrenamiento por aprendizaje de máquina u otras técnicas de control.

En [18] se desarrollan métodos difusos para lograr la estabilización y control del péndulo invertido rotatorio de la empresa Quanser®, que consiste en un sistema mecánico de péndulo simple conectada a un servomotor rotativo que se encarga de mantener el péndulo en una posición estable. Presenta dos sistemas de control, donde el primero de ellos se enfoca en el swing up visualizado al momento en que el péndulo se mueve hace arriba desde la posición vertical hacia abajo, hasta la posición vertical inestable hacia arriba. El segundo sistema se encarga de la estabilización en este punto inestable. Las reglas para el swing up están escritas heurísticamente de tal manera que cada swing produce una mayor acumulación de energía. La estabilización se logra mediante el mapeo de una ley de control de estabilización LQR a dos motores de inferencia difusa, lo que reduce la carga computacional en comparación con el uso de un único motor de inferencia difusa. La robustez del control de equilibrio se prueba colocando una botella de agua en la punta del péndulo.

En [19] se busca la implementación de controladores oscilantes, de conmutación y estabilizadores para péndulo rotativo invertido. Se emplea un método para la obtención de la energía necesaria para elevar el péndulo y en un controlador por retroalimentación de estado mixto para la estabilización del péndulo invertido en su punto a estabilizar, usando el péndulo invertido rotativo de la empresa Quanser®.

En [20] se presenta un enfoque sobre el modelado de Bond Graph aplicado a un experimento sobre el péndulo rotativo invertido de Quanser® y seguidamente del cálculo del modelo matemático. El artículo muestra la composición de todo el sistema y el software usado para lograr el modelado del péndulo rotativo invertido. El sistema se divide en subsistemas que se modelaron de manera separada, de esta forma se logran obtener submodelos y a partir de esto se construye el modelo general del sistema. Muestra el modelo matemático general de la planta del péndulo rotativo invertido, se construye y se simula usando el entorno BondSim.

En [21] se muestra un laboratorio implementado por la empresa Quanser® para un péndulo invertido, el laboratorio se implementa desde el modelado del péndulo invertido hasta el diseño del control de este. El laboratorio muestra el modelo matemático del péndulo invertido donde se obtienen todas las ecuaciones no lineales y se muestra su linealización a partir del modelo lineal de espacio de estados.

Los modelos expuestos en [19], [20], [21] y en los anteriores artículos, donde no se presentan sistemas de control basados aprendizaje de máquina, toman de base el modelo de estados de la planta los cuales representan sistemas no lineales con incertidumbres, lo que conlleva a que el modelo matemático resultante no sea preciso.

En [22] realizan un modelo de predicción de tendencias para ayudar al gobierno de Estado Unidos a tomar medidas sanitarias sobre la tendencia que lleva de la pandemia registrada. Para su desarrollo utilizan algunos modelos de regresión y modelos de redes neuronales para predecir datos pandémicos en este país. Este trabajo toma como métodos la regresión lineal, la regresión logistica y la implementación de una red neuronal recurrente RNN para predecir los casos por millón de personas. Adicionalmente, toma estos tres métodos implementados y proceden a una evaluación de estos por medio del error cuadrático medio (MSE). Al realizar la evaluación de estos métodos, la conclusión dada por los autores conllevaba a la elección de una red neuronal recurrente para este tipo de casos y este tipo de sistemas, puesto que, aunque su tiempo de aprendizaje sea más largo, lo que lo hace al sistemas más lento que los implementados por los métodos de regresión en cuanto tiempo de inferencia, este era más preciso, esto al observar que el error cuadrático medio era menor que el de los otros métodos.

En [23] se usan métodos de regresión lineal multiple, regresión logarítima, la regresión polinomial y el método de una red neuronal, para así lograr crear modelos de pronóstico para la generación de energía solar. Estos modelos de pronósticos son comparados entre sí a través del error del sistema para lograr encontrar aquel porcentaje de error mínimo que garantice un pronostico sobre la generación solar apropiado. En este artículo se concluye que a partir de la red neuronal se pudo mapear la mayor cantidad de datos convirtiendola en el método más preciso para el desarrollo del modelo de pronóstico de generación solar.

En [24] se busca realizar una comparativa en rendimiento entre un modelo líneal y un modelo de red neuronal para un sistema no lineal para la predicción de temperatura. Este documento se centra en la métrica de error cuadrático medio MSE. En este trabajo se logra concluir que la inclusión de una red neuronal para la predicción de un sistema no lineal, presenta un mejor desempeño con respecto a su error cuadrático medio, haciendo mejor opción sobre un modelo de regresión lineal.

En [25] se observa una simulación la cuál utiliza un control neuronal inverso, con el fin de controlar la corriente en un convertidor Buck. También se presenta la implementación de un controlador de módulos neuronales en paralelo con el fin de mejorar el desempeño del controlador inverso y de evaluar la viabilidad del uso de este tipo de control neuronal para su posterior implementación física.

En [26] se propone un modelo de predicción de lluvia basado en la arquitectura de redes neuronales NARX, entrenado a partir del algoritmo de autoadaptación SALM (Self Adaptative Levenberg-Marquardt), usado para la selección de pesos de forma más eficaz, el cual es una optimización del algoritmo Levenberg-Marquardt tradicional. Como métricas de evaluación el error cuadrático medio (MSE por sus siglas en inglés) y diferencia porcentual de la raíz media cuadrática (PDR por sus siglas en inglés). La comparación de esta arquitectura se realiza con la Levenberg-Marquardt, el modelo de regresión lineal y otros dos modelos como el Perceptrón híbrido Wavelet-Season-Multilayer (W-SAS-MP), y un modelo de optimización híbrida adaptativa de Particle Swarm Optimization (PSO). Al realizarse la comparación con 4 conjuntos de datos diferentes, concluye que las arquitecturas que presentaron mejor desempeño, según las métricas de evaluación, son las basadas en la redes NARX entrenadas con los algoritmos de Levenberg-Marquardt.

El uso del aprendizaje de máquina para el control y estabilización del péndulo invertido busca implementar sistemas basados en la planta y no en su modelo de estados, tratando de obtener una mejor precisión, o, en su defecto, a partir de este modelo no lineal obtenido implementar

por medio de técnicas de aprendizaje de máquina un modelo que sea capaz de aprender la dinámica de la planta y así complementar el modelo no lineal ya obtenido.

Capítulo 3

Justificación

El sistema de péndulo invertido ha sido de gran ayuda para poder entender el comportamiento en robots capaces de mantenerse en equilibrio, al igual que en enfoques relacionados al transporte aéreo [27]. Con la exploración de nuevas técnicas de control por aprendizaje de máquina u otros controles un poco más avanzados, se logra dar una visión de cómo estos pueden llegar a trabajar sobre sistemas no lineales, tal como el péndulo invertido simple y ayudar a solucionar dicho problema de linealidad con mayor facilidad a los controles convencionales [4].

Se espera que con la implementación de un control por aprendizaje de máquina aplicado sobre un péndulo invertido simple, en una plataforma simulada, se pueda a futuro abrir la posibilidad de trabajar con otros sistemas no lineales, como sistemas de péndulos dobles y así mismo contribuir con la investigación de nuevos controladores usando nuevos algoritmos que ponga en práctica técnicas de aprendizaje de máquina o redes neuronales, que serán de beneficio para el estudio de la robótica y la automatización.

Los motivos que llevaron a la investigación de un control de péndulo invertido por medio de técnicas de aprendizaje máquina, son debido a que los controles generalmente usados para un sistema no lineal son tan complejos de linealizar como de controlar e implementar, por lo tanto, se busca facilitar este proceso con la técnica ya mencionada, al finalizar se busca mejorar la estabilización de un péndulo invertido en una plataforma simulada en el Software Simulink.

Este proyecto tiene como finalidad el diseño de un sistema de control para la estabilización de un péndulo invertido simple utilizando controladores basados en aprendizaje máquina. Este proyecto será implementado en una plataforma simulada en el software Simulink, la cual

permite observar una animación del comportamiento del péndulo invertido de manera gráfica, desarrollada por el docente del Departamento de Ingeniería Mecánica de la Universidad Detroit, Rick Hill [28].

Capítulo 4

Impacto Social

Observando el avance industrial nacional y local, se encuentra que los estudios en las técnicas usadas para el desarrollo de estos avances también ha presentado un gran crecimiento. Algunas de las áreas de estudio e investigación que se ven aplicadas en la industria van desde técnicas de control, inteligencia artificial o automatización. Con todo esto, se busca el mejoramiento y refuerzo en la enseñanza en estas áreas desde los campos universitarios. A partir de este proyecto, se desean reforzar las bases que sirvan de referencia para el estudio de controles enfatizados en la investigación, integración e implementación de técnicas como la inteligencia artificial, el aprendizaje de máquina y el control moderno que puedan llegar a dar solución a problemáticas en la vida cotidiana a bajo costo para así poder llegar a portar en el desarrollo de enseñanzas en estas áreas en las aulas de clases de los claustros universitarios.

La proyección social de este proyecto es brindar la posibilidad de estudiar otras técnicas y arquitecturas de control para estabilizar sistemas no lineales como el péndulo invertido que son estudios para la construcción de sistemas industriales, sistemas robóticos, sistemas de control, y brindar la posibilidad de agregar otras funcionalidades a estos. A partir de este estudio, se podría llegar a ahorrar tiempo y costos, ya que los sistemas que son usados generalmente para estabilizar péndulos y otros sistemas no lineales son muy complejos, además de presentar, en la mayoría de casos, un elevado costo al momento de implementarlos.

Capítulo 5

Objetivos

5.1. Objetivo general

Implementar un sistema de control para un péndulo invertido simple que permita estabilizar la posición angular al rededor de un punto de trabajo específico, basado en técnicas de aprendizaje de máquina.

5.2. Objetivos específicos

- Seleccionar un método de aprendizaje de máquina para el control de posición angular para el péndulo invertido, a partir de la revisión del estado del arte.
- Validar el método de aprendizaje de máquina seleccionado para el control de la posición angular del péndulo invertido simple, en una plataforma simulada en el Software Simulink.
- Seleccionar métricas que permitan la validación del sistema de control propuesto, a partir de una revisión del estado del arte.
- Comparar el desempeño del sistema de control para una referencia de posición angular predeterminada simulado en el objetivo específico 2, con algún método de control clásico simulado empleando el mismo modelo de la planta, utilizando las métricas previamente seleccionadas.

Capítulo 6

Marco Teórico

Es importante tener claridad en algunos conceptos que servirán para tener mayor entendimiento del proyecto. Algunos de los conceptos se enuncian a continuación:

Péndulo Invertido Simple: Es un tipo de sistema que ha sido utilizado en los estudios de control de sistemas no lineales, esto al ser un sistema altamente no lineal. A partir de esto, lo que se busca es localizar el péndulo de manera vertical, es decir a $180°$ de su región de reposo y desde allí aplicar el control diseñado para que este se siga manteniendo estabilizado en esta región de trabajo. Para empezar a trabajar en el diseño de la red neuronal, primero se debe observar el modelo no lineal de la planta, ya que a partir de este, es decir, de la dinámica del sistema, se puede entrenar dicha red para que su salida sea lo más parecida a la referencia deseada. En la figura 1 se puede apreciar el esquemático del sistema del péndulo invertido.

El modelo no lineal del péndulo invertido simple se da a continuación:

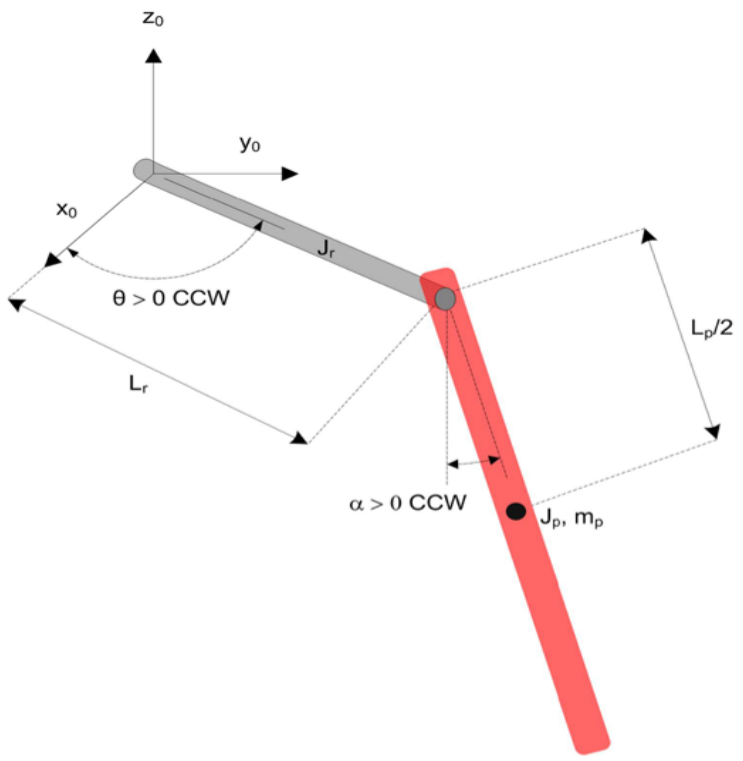

FIGURA 1: Esquemático del sistema del péndulo invertido [28].

Siendo θ el ángulo del brazo giratorio y α el ángulo del péndulo. También es importante dejar claro que cuando el brazo giratorio y el péndulo giran en el sentido contrario a las agujas del reloj, el valor de estos ángulos, al igual que el par aplicado del motor (τ) tomarán valores positivos.

Con lo anteriormente dicho, se tienen las siguientes ecuaciones del movimiento del péndulo.

Primera ecuación:

Donde m_p es la masa del péndulo, L_r la longitud total del brazo giratorio, L_p la longitud total del péndulo, α el ángulo que genera el péndulo, J_r momento de inercia del brazo giratorio sobre el centro de masa y θ representa el ángulo generado por el brazo giratorio.

$$A = (m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 \cos^2(\alpha) + J_r)\ddot{\theta} \tag{6.1}$$

$$B = (\frac{1}{2} m_p L_p L_r \cos(\alpha))\ddot{\alpha} \tag{6.2}$$

$$C = (\frac{1}{2} m_p L_p^2 \sin(\alpha)\cos(\alpha))\dot{\theta}\dot{\alpha} \tag{6.3}$$

$$D = (\frac{1}{2} m_p L_p L_r \sin(\alpha))\dot{\alpha}^2 \tag{6.4}$$

Siendo esta la primer ecuación de movimiento:

$$A - B + C + D = \tau - \beta_r \dot{\theta} \tag{6.5}$$

Siendo τ el torque o par aplicado en la base del brazo giratorio generado por el servomotor conectado al brazo y β_r el coeficiente de amortiguación viscoso del brazo giratorio.

Segunda ecuación:

$$A = \frac{1}{2} m_p L_p L_r \cos(\alpha)\ddot{\theta} \tag{6.6}$$

$$B = (J_p + \frac{1}{2} m_p L_p^2)\ddot{\alpha} \tag{6.7}$$

$$C = \frac{1}{4} m_p L_p^2 \cos(\alpha)\cos(\alpha)\dot{\theta}^2 \tag{6.8}$$

$$D = \frac{1}{2} m_p L_p g \sin(\alpha) \tag{6.9}$$

Siendo esta la segunda ecuación de movimiento:

Donde β_p el coeficiente de amortiguación viscoso del brazo péndulo.

$$A + B - C + D = -\beta_p \dot{\alpha} \tag{6.10}$$

A partir de estas ecuaciones y dejando claro que estás ecuaciones representan el sistema no linealizado, se puede empezar a trabajar en el diseño de la red neuronal, que aprenda la dinámica del sistema e intente estabilizar el péndulo sin necesidad de recurrir a una linealización del sistema.

Redes Neuronales: Se pueden describir como un modelo inspirado en el comportamiento y funcionamiento del cerebro humano. Su estructura interna se centra en un conjunto de nodos interconectados transmitiendo señales entre sí, denominados neuronas artificiales. Estas redes poseen una señal de entrada que se transmite a través de las neuronas artificiales hasta generar una salida.

Las redes neuronales presentan como objetivo principal el aprender modificándose automáticamente a sí mismo a tal punto de poder llegar a realizar tareas que aplicando las reglas de programación clásicas no podrían realizarse. A este punto, se puede decir que el entre las funciones principales de las redes neuronales es llegar a automatizar funciones que en un principio sólo podrían ser realizadas por personas.

¿Cómo funcionan?

Como se mencionó anteriormente, una red neuronal es un conjunto de neuronas artificiales conectadas entre sí, estas neuronas se agrupan formando capas que son la estructura general de la red neuronal. Las neuronas poseen un peso (valor numérico) que puede modificar la entrada recibida que, a su vez, se convierte en una nueva entrada a una nueva capa de neuronas donde se vuelve a modificar o procesar para dar paso capa por capa hasta completar la red. Una vez alcanzado el final de la red, se obtiene una salida que será la predicción calculada. Entre más capas que posea una red neuronal y más compleja sea, también será más compleja las funciones que pueden llegar a realizar. La estructura de una red neuronal está dada esquemáticamente por la figura 2:

La capa de entrada es aquella etapa inicial de la red neuronal a la que se conectan las entradas reales del sistema con su respectivos pesos, los cuales serán usado para modificar esta entrada e ingresarla seguidamente a la capa oculta, la cual, recibe este nombre debido a que el valor proveniente de las salidas de la Capa de Entrada, ingresan como entrada a esta capa, son desconocidos, al igual que el valor que sale de esta. Seguido de la capa oculta, sigue la capa de salida, que es la última capa de la red y presenta el resultado visible de la red, el valor predicho de esta.

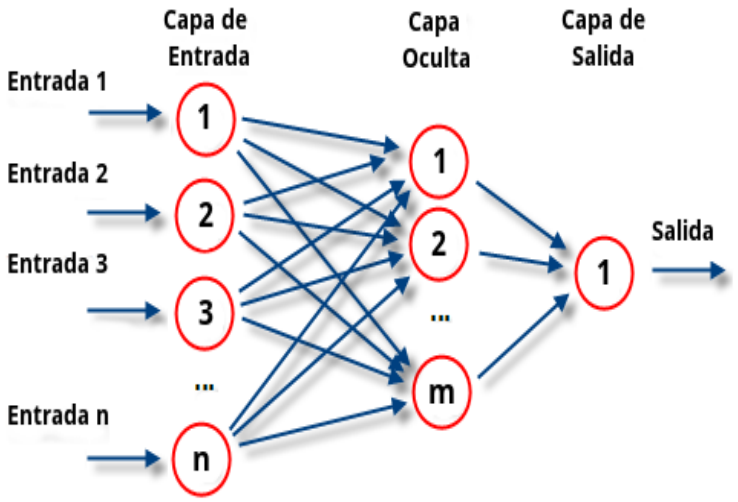

FIGURA 2: Esquemático de una red neuronal [29].

Para que las redes neuronales funcionen es necesario realizar un proceso de entrenamiento. Este entrenamiento se realiza modificando el valor de los pesos en las neuronas para que esta intente extraer los resultados que se esperan. Esto se consigue introduciendo datos de entrenamiento en la red neuronal y en función del resultado obtenido, se modifican los pesos de las neuronas según el error obtenido y en función de cuánto haya contribuido cada neurona a dicho resultado. De esta forma se consigue que la red neuronal pueda aprender, llegando a obtener un modelo que sea capaz de proporcionar resultados muy acertados incluso cuando los datos que se van introduciendo son diferentes a los que se usaron en su proceso de entrenamiento. Este método se conoce como BackPropagation. Este método ha permitido realizar grandes optimizaciones para cálculos de gran costo computacional.

Con la capacidad de las funciones de las redes neuronales, se puede llegar a aproximar cualquier tipo de función existente con el entrenamiento suficiente. En esencia, las redes neuronales se utilizan en tareas de predicción y clasificación, aunque su rango de uso actualmente es amplio y de gran utilidad [29].

Algoritmo BackPropagation: BackPropagation es uno de los algoritmos más utilizado para el entrenamiento de redes neuronales multicapa. Permite el cálculo del gradiente en cualquier punto del dominio de la función de coste, ayudando a obtener el cálculo del mínimo de esta función, que en el ultimas, este es el objetivo del entrenamiento. Todo esto lo hace a partir de un calculo matemático arduo, que en en términos generales, consiste en ir hacia atrás en la red neuronal calculando de las derivadas parciales del error con respecto a todos los pesos y bias de las neuronas. Una vez calculadas estas derivadas parciales, se usa su resultado por medio del

algoritmo de gradiente descendiente para intentar minimizar el error a través del ajuste de los pesos en cada una de las neuronas empezando desde la ultima capa hacia la primera (Figura 3).

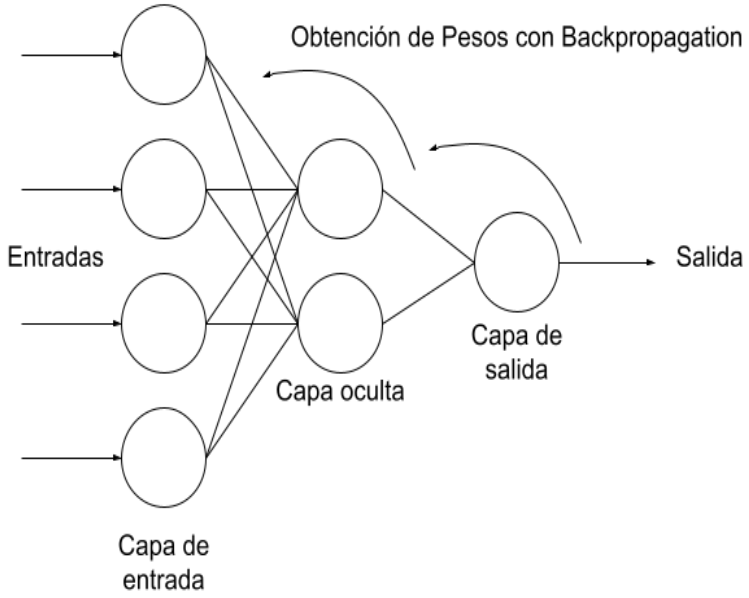

FIGURA 3: Esquemático de una red neuronal con backpropagation [30].

Para entender un poco mejor el comportamiento del algoritmo Backpropagation, se muestran las ecuaciones de una red neuronal sencilla aplicando backpropagation, como se muestra en el ejemplo de la figura 4.

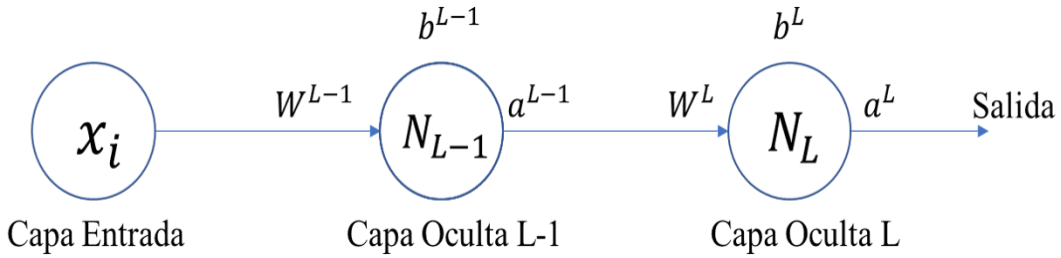

FIGURA 4: Estructura de una red neuronal sencilla con backpropagation [31].

Siendo L la capa oculta y L-1 la primera capa. Con esto, se puede determinar que el comportamiento de la neurona N_L viene dado por el peso (W^L) que está sujeto al enlace que une esta neurona con la neurona anterior (N_{L-1}) y por el bias añadido, b^L.

Siguiendo lo mencionado anteriormente, se desea calcular la derivada parcial de la función de coste con respecto a los parámetros del peso y el bias, tal como se aprecia en las expresiones 6.11.

$$\begin{matrix} \frac{\partial C}{\partial W^L} \\ \frac{\partial C}{\partial b^L} \end{matrix} \tag{6.11}$$

En estas derivadas parciales, la función de Coste C, es la resultante después de analizar el comportamiento de la red neuronal con todas las muestras del conjunto de entrenamiento. Si se quiere tomar una muestra i-esima, las anteriores expresiones estarían dadas por las expresiones 6.13.

$$\begin{matrix} \frac{\partial C_i}{\partial W^L} \\ \frac{\partial C_i}{\partial b^L} \end{matrix} \tag{6.12}$$

Donde se analizar que el coste C se puede calcular como el valor medio de los valores C_i que se estiman para todas las muestras del conjunto de entrenamiento. Con esto, se puede suponer que la función de coste está dada por el error cuadrático medio.

$$C_i = (y - p_i)^2 \tag{6.13}$$

Donde **y** representa el valor de la salida que se espera de la red neuronal para una muestra i-esima y p_i representa el valor real devuelto por la red neuronal en esta muestra i-esima. El valor de p_i viene dado por la multiplicación del valor proveniente de la neurona anterior (a^{L-1}) por el peso del enlace (W^L), más el valor del bias (b^L). Una vez calculado este valor, se pasa como parametro de la función de activación de la red, σ. Tal como se aprecia en la expresión 6.14

$$p_i = \sigma(W^L * a^{L-1} + b^L) \tag{6.14}$$

Teniendo así que la función de coste puede ser escrita como se muestra en la expresión 6.15.

$$C_i = (y - \sigma(W^L * a^{L-1} + b^L))^2 \tag{6.15}$$

Por temas prácticos z^L estará dada por la combinación lineal $W^L * a^{L-1} + b^L$. De modo que:

$$p_i = \sigma(z^L) \tag{6.16}$$

A partir del cálculo infinitesimal, lo cual dice que la derivada parcial de la función de coste C_i con respecto a wL, es igual a la derivada parcial de C_i con respecto a σ, multiplicado por la derivada de σ respecto de z^L, multiplicado por la derivada de z^L con respecto a W^L. Esto es lo que se conoce regla de la cadena y se aprecia en la expresión 6.17.

$$\frac{\partial C_i}{\partial W^L} = \frac{\partial C_i}{\partial \sigma}\frac{\partial \sigma}{\partial z^L}\frac{\partial z^L}{\partial W^L} \tag{6.17}$$

Al calcular las derivadas parciales se da como resultado:

- La derivada parcial de C_i con respecto a σ es igual a $2(y - \sigma)$.
- La derivada parcial de σ con respecto a z^L es la derivada de la función de activación con la que se está trabajando.
- La derivada parcial de z^L con respecto a W^L es a^{L-1}.

Del mismo modo, la derivada parcial de C_i con respecto a b^L está dada por la expresión 6.19.

$$\frac{\partial C_i}{\partial b^L} = \frac{\partial C_i}{\partial \sigma}\frac{\partial \sigma}{\partial z^L}\frac{\partial z^L}{\partial b^L} \tag{6.18}$$

Donde se puede apreciar que el resultado de las derivadas parciales C_i con respecto a σ y de σ con respecto a z^L son las mismas obtenidas en la expresión 6.17. La diferencia se aprecia en el cálculo de la derivada parcial de z^L con respecto a b^L, la cual es igual a 1.

Con la regla de la cadena se puede apreciar el calculo de atrás hacia delante, de las derivadas parciales de la función de coste con respecto a los pesos y bias de la red neuronal. Mostrando que el cálculo de la derivada parcial de la siguiente capa oculta, hará uso de la derivada parcial calculada en la capa anterior, haciendo que el cálculo del gradiente sea relativamente más sencillo.

Como se muestra en las ecuaciones anteriores, el algoritmo backpropagation permite el cálculo del gradiente en cualquier muestra i-esima del dominio de la función de coste, permitiendo, además, el cálculo del mínimo de esta función, siendo este el objetivo del entrenamiento [31].

Redes Neuronales Recurrentes : Para entender lo que representa una red neuronal recurrente, es importante saber que también existen otro gran grupo de redes, denominadas Redes Multicapa de alimentación hacia adelante o redes feed-forward [32]. Este tipo de redes se denominan estáticas ya que producen una única salida para un conjunto de entrada, esto se debe a su naturaleza unidireccional, donde se organizan en capas conectadas de manera unidireccional, haciendo que el estado de una red sea independiente del estado anterior.

Con lo anteriormente mencionado, se puede decir que una red recurrente o red RNR se representa como un sistema dinámico donde el cálculo de una entrada, en un paso, depende del paso anterior, es decir, cuenta con realimentación. A partir de este concepto, las redes RNR son capaces de realizar una gran diversidad de tareas computacionales que van desde el tratamiento de secuencias, la predicción o continuación de una trayectoria, predicciones no lineales hasta la modelación de sistemas dinámicos. Son conocidas como redes espacio-temporales o dinámicas.

Con este tipo de redes existen tres casos de tareas esenciales que se pueden ejecutar: Reconocimiento de secuencias: A partir de una RNR, se produce un patrón de salida cuando se especifica una secuencia de entrada; Reproducción de secuencias: La RNR debe ser capaz de generar el resto de una secuencia cuando ve parte de esta; Asociación temporal: A partir de la RNR, una secuencia de salida se debe producir en respuesta a una secuencia de entrada específica [32].

La representación básica de una red RNR se presenta en la figura 5.

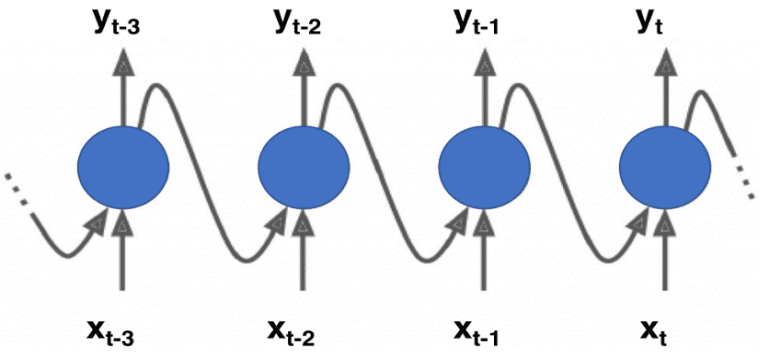

FIGURA 5: Esquemático de una red neuronal recurrente [33].

Donde en cada instante de tiempo, cada neurona recibe dos entradas, puede tener la opción de recibir la entrada de la capa anterior y a su vez la salida del instante anterior de la misma capa [33].

A continuación, en la expresión 6.19 se presenta la ecuación que describe el comportamiento de una Red Neuronal Recurrente.

$$y(t) = f(Wx_t + Uy_{t-1} + b) \tag{6.19}$$

Donde $x = (x_1, \ldots, x_T)$ representa la secuencia de entrada que proviene de la capa anterior, W representa los pesos de la matriz y b representa el valor de los bias. Las redes neuronales recurrentes extienden la función $y(t)$ con una conexión recurrente en el tiempo donde, U representa la matriz de pesos que opera sobre el estado de la red en un instante de tiempo anterior (y_{t1}) [33]. Como se menciona anteriormente, en la fase de entrenamiento a través de Backpropagation también se actualizan los pesos de esta matriz aplicando las derivadas parciales mencionadas en este algoritmo.

Tipos de redes neuronales recurrentes:

Redes recurrentes Simples -SRN o Elman: Este tipo de red se basa en la estructura simple mostrada en la figura 5, por tanto, la ecuación que representa el comportamiento de esta red es la mostrada en la expresión 6.19. Como se mencionó anteriormente, se diferencia de las demás redes por poseer un sistemas de realimentación que le permite tener memoria o lo que se denomina "memory cell". Gracias a esta memoria interna, las redes recurrentes pueden recordar información importante de la entrada que recibieron ayudándoles a ser más precisas en la predicción.

Redes LSTM: *L*as redes LSTM (Long Short Term Memory) están compuestas por unidades LSTM. Se usan para solventar el problema que una red recurrente convencional presenta en su entrenamiento, donde los gradientes retropropagados tienden a crecer o a desvanecerse con el tiempo, debido a que el gradiente no solamente depende del error presente sino también de los errores pasados, al acumular estos errores, se provocan errores para memorizar dependencias a largo plazo.

La red LSTM solventa este problema con puertas que se encargan de controlar el modo en que la información fluye dentro o fuera de la unidad [34].

- La puerta de entrada controla cuando la información nueva puede entrar en la memoria.
- La puerta del olvido controla cuando se olvida una parte de la información, permitiendo a la celda separar entre datos importantes y datos superfluos, dando la posibilidad de dejar sitios para nuevos datos
- La puerta de salida controla cuando se utiliza en el resultado de los recuerdos almacenados en la celda. En la figura 6 se aprecia la estructura de una red LSTM.

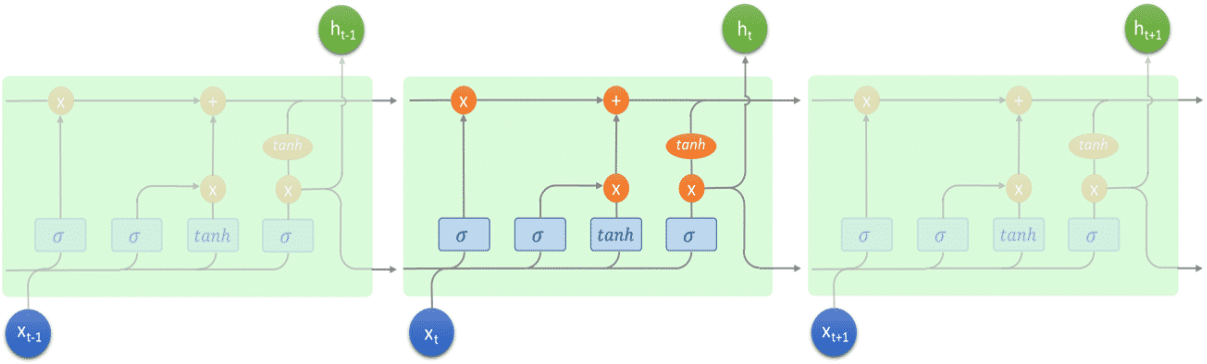

FIGURA 6: Estructura de una red LSTM [35].

Redes Neuronales NARX: (Nonlinear Autoregressive with eXogenous input) es una red con un modelo auto regresivo no lineal con entrada exógena, y es un tipo de red recurrente con conexiones de retroalimentación que encierran diversas capas de la red. Se usa comúnmente en el modelado de series de tiempo. Con la capacidad de aceptar entradas dinámicas, es decir, discretas o continuas). Las redes neuronales NARX tienen la propiedad de combinar la estructura de las redes NAR y de las redes FTDNN que en esencia son redes con retrasos en el tiempo, estos retrasos son incluidos tanto a la entrada como en las respuestas de la propia red neuronal, funcionando con un modo de realimentación. Con esto, las redes NARX pueden predecir

valores futuros de una seria a partir de los valores anteriores, por lo que son un tipo de red neuronal que se adapta bien al modelado de sistemas dinámicos [36].

La estructura de una red NARX se presenta en la figura 7.

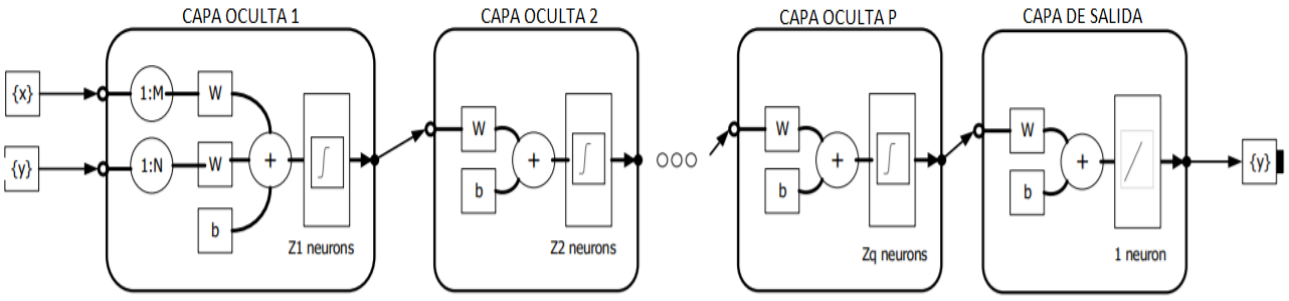

FIGURA 7: Esquema de una red NARX [36].

Donde se aprecia que la señal de salida se usa también con entrada del sistema retardada N veces. Las entradas está retardada M veces.

Las redes NARX presentan una ventaja al incluir la predicción de un periodo, es de decir, la salida del sistema 'y' en un tiempo t, como dato de entrada y como períodos siguientes, haciendo que esta red pueda resumir información de las variables exógenas, permitiendo la inclusión de menos retrasos de estas, lo que conlleva a resumir el número de parámetros a estimar.

A continuación se presenta la expresión matemática que representa la red NARX:

$$y(t) = f(\{y\}, \{x\}) f((y(t-1), y(t-2), ..., y(t-N), x(t), x(t-1), ...x(t-M)) \tag{6.20}$$

Donde (f) representa una función desconocida que se desea obtener al usar la esta forma de identificación; teniendo la entrada {x} y las salidas anteriores del sistema {y}, donde y(t) representa la respuesta que el sistema predijo en el instante actual (t), {y} es el vector que representa la salida anterior del sistema retardada N instantes de tiempo, el vector {x} representa la entrada del sistema en el instante (t) y las entradas anteriores retardadas M instantes de tiempo. Basados en la figura 7, se observa que la red NARX es equivalente a un sistema perceptrón multicapa. Teniendo en la entrada tantas neuronas como datos de entradas se usen y considerando que los datos de entrada son las M entradas anteriores y la N salidas anteriores del sistema.

Control Neuronal por modelo Inverso: Con esta técnica se busca cancelar la dinámica de la planta, eso se hace agregando una red neuronal que funcionará a partir de una aproximación matemática del inverso de la planta, esto hace que la salida sea lo más parecida posible a la

referencia, para el caso de este proyecto se está hablando del valor dado por el valor del ángulo alfa del péndulo, es decir el valor en el que se encuentra estabilizado en la región de trabajo, más exactamente en el valor de pi.

Con este método de control se tienen dos posibilidades o tendencias, una de ella consiste en la utilización de un set de datos previamente obtenidos para el entrenamiento de la red neuronal y seguidamente ser usada esta red como controlador puesto que en este caso se está cancelando la dinámica de la planta tal como se muestra en la figura 8.

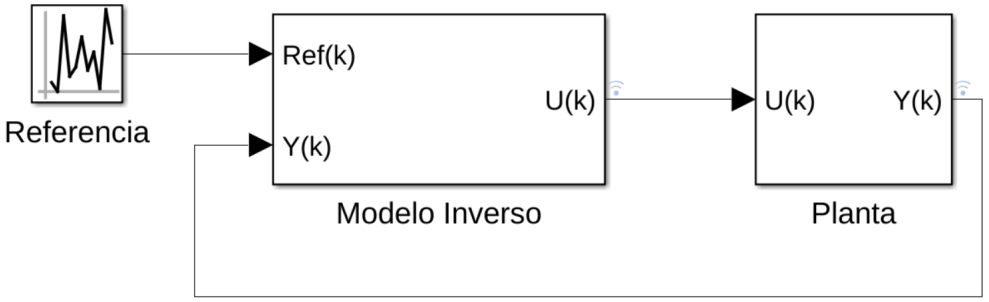

FIGURA 8: Esquema de Control por Modelo Inverso [37].

Control LQG: (Linear Quadratic Gaussian) Se establece como una combinación entre una ganancia de realimentación de estados y un estimador de estados. Este control toma de base el modelo lineal del sistema y utiliza modelos de ruido estocásticos, es decir, ruidos de medición y perturbaciones del sistema.

Descenso de gradiente: Es uno de los algoritmos de optimización más usados en aprendizaje automático, más específicamente en el amplio uso de las redes neuronales. Gracias a este, se puede minimizar cualquier tipo de función, ya que gracias a su cálculo, se logra ajustar los parámetros de la red de tal forma que se logre minimizar la desviación a la salida de la red.

Existen diferentes formas de aplicar el descenso de gradiente:

- **Descenso de gradiente en lotes (bash):** Esta forma toma todos lo datos disponibles y se introducen una vez. Lo que podría llegar a suponer inconvenientes ya que al usar todos los datos de muestra, llegará al punto en que las variaciones serán mínimas, llevando a un problema de estancamiento.
- **Descenso de gradiente estocástico:** A diferencia de la anterior forma, el descenso de gradiente estocástico evita el estancamiento, ya que este introduce una única muestra aleatoria en cada iteración, calculando el gradiente para esta muestra en particular. El problema

de esta forma se evidencia a su lentitud ya que tendrá que hacer un gran número de iteraciones para el calculo del gradiente.

- **Descenso del gradiente (estocástico) en mini-lotes (o mini-batch):** Por medio de esta forma se permite introducir un grupo de N muestras por cada iteración, ayudando a evitar el problema de estancamiento presente en la primera forma y ayudando a la red a tardar menos tiempo solventando un poco el problema mostrado en la segunda forma [38].

Levenberg Marquardt (LM): Es un método iterativo usado para resolver problemas no lineales de estimación de parámetros usando el método de mínimos cuadrados que, generalmente, se presentan en sistemas de ajuste de curvas. El algoritmo al ser iterativo, el vector de actualización de la solución en cada iteración realizada, se realiza a través de una combinación entre el algoritmo Gauss - Newton y el método de descenso de gradiente. Ante esto, cuando la solución actual se encuentra lejos de un mínimo local, el algoritmo LM se comporta como el método de descenso de gradiente, esto podría ser una solución lenta, pero con una gran precisión. Cuando la solución se encuentra cercana al mínimo local, el algoritmo se comporta como un algoritmo Gauss-Newton. Esta presenta una convergencia más rápida. El uso del algoritmo LM surge como una alternativa para evitar los problemas presentes en el algoritmo Gauss Newton.

Al momento de implementar el algoritmo Levenber Marquardt, se debe tener presente que al ser usado en la solución de problemas no lineales de estimación que, a su vez, involucra un gran número de parámetros desconocidos, su tiempo de ejecución es un poco más lento que muchos otros algoritmos, debido al cálculo de la matriz de sensibilidad. Sin embargo, el método Levenberg Marquardt usado como entrenamiento de redes neuronales tiende a tener resultados de entrenamiento más precisos frente a otros métodos de entrenamiento. La diferencia respecto a otras técnicas de entrenamiento se refleja en que Levenberg Marquardt tiende a necesitar más memoria por la cantidad de iteraciones realizadas, haciendo también el método mas lento [39].

Error cuadrático medio: Es uno de los criterios de evaluación que más se usan a la hora de abordar problemas de regresión. Su uso más común es para tratar temas con aprendizaje automático supervisado [40]. Su funcionamiento consiste en medir cuánto error existe entre dos conjunto de datos, esto, comparando el valor del valor predicho y el valor esperado o conocido. Este criterio ayuda a definir la precisión que existe en un sistema de redes neuronales, por lo que la convierte en una de los criterios de evaluación más importantes. La ecuación matemática del este criterio está dada por la siguiente expresión:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Y}_i - Y_i)^2 \tag{6.21}$$

Donde $\widehat{Y}$ representa un vector n de predicciones y Y la salida esperada o conocida.

Regresión Lineal: Es utilizada para la predicción del valor de una variable respecto a otra variable. Donde, la variable la cual se espera predecir se asigna a la variable dependiente mientras que la variable usada para predecir se llama variable independiente. Esta manera de analizar el comportamiento de datos predichos funciona a partir de la estimación de coeficientes en la ecuación lineal (en el uso de redes neuronales estos coeficiente se llaman pesos) involucrando una o más variables independientes (o entradas) que mejor predicen el valor de la variable independiente. Esta forma usa una linea recta ajustada en una superficie de tal forma que elimine la mayor cantidad de distancia entre los valores de salida previstos y los valores conocidos. Usa criterios como el error MSE para predecir dichas variables [41].

Identificación de Sistemas: Se permite hacer una identificación de sistemas a partir de la construcción de modelos y funciones matemáticas de los sistemas dinámicos, esto de las entradas del sistema y a partir de ellas observar el comportamiento de la salida. La identificación de sistemas puede llegar a usar desde la inferencia de fuerzas, masas o momentos que son incitados sobre el sistema dinámico a identificar. A partir de las definiciones mencionadas anteriormente, la identificación de sistemas puede llegar a usar métodos de regresión lineal y no lineal pues que la mayoría de veces, esta identificación se presenta para problemas de control. Es importante saber que para la identificación del sistema, se debe simular el comportamiento real de este en los casos que exista o se conozca conocimiento previo limitado de la estructura del sistema. En el caso del péndulo invertido, con la identificación del sistema se debe intentar simular el comportamiento del péndulo para que a partir de eso, se pueda implementar las posibles técnicas de control sobre el péndulo. La identificación de sistemas no lineales con lleva a una complejidad mayor, puesto que allí se debe realizar una selección de la estructura del modelo con un cierto número de parámetros y también la selección de un algoritmo que sea capaz de estimar estos parámetros [42].

Teorema de Muestreo de Nyquist: Este teorema explica la relación existente entre la velocidad de muestreo y la frecuencia de la señal, afirmando que la velocidad de muestreo f_s debe ser mayor que el doble del componente interés de frecuencia más alto en la señal medida, así cómo se observa en la ecuación 6.22. A esta frecuencia se le llama frecuencia Nyquist (f_N) [43].

$$f_s > 2 * f_N \tag{6.22}$$

A partir del teorema de Muestreo de Nyquist plantea que una velocidad de muestreo baja podría causar una reconstrucción inexacta de la señal [43].

Capítulo 7

Desarrollo Metodológico

Se explicará el desarrollo del proyecto que busca mostrar el cumplimiento de los objetivos planteados en este, dividido en 8 etapas reflejadas en la figura 9, para hacer más sencillo su entendimiento. La investigación del proyecto es de tipo cuantitativa y experimental puesto que se realiza un estudio desde el estado del arte, y a partir de su análisis se pueda realizar el diseño del controlador con las respectivas pruebas y correcciones.

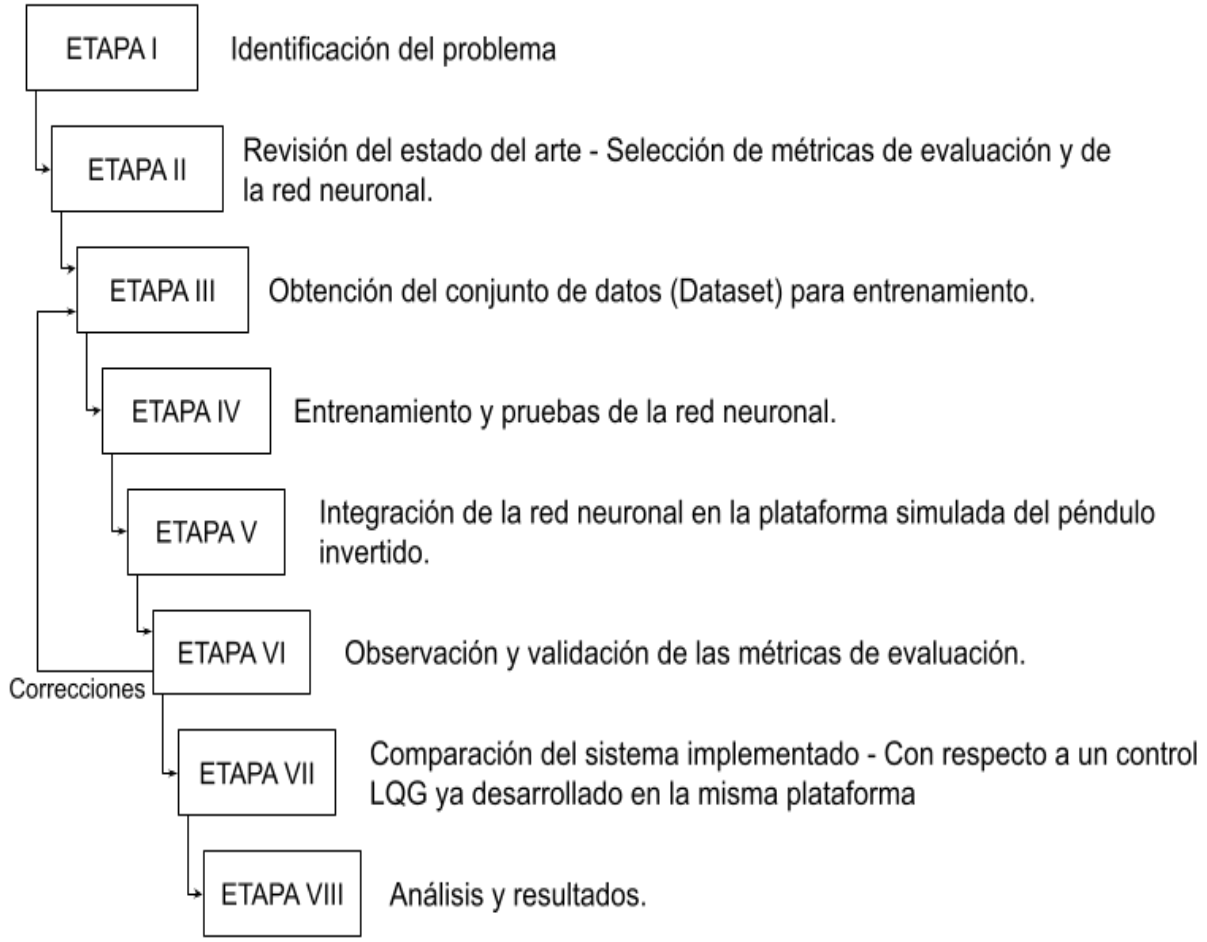

FIGURA 9: Desarrollo Metodológico por etapas (Fuente: Autores).

7.1. Etapa I: Identificación del Problema

En esta etapa se busca identificar las variables de entrada y salida del péndulo, el rango de trabajo en que trabajará el controlador y la señal que se quiere controlar.

7.2. Etapa II: Selección de las métricas de evaluación y la red a utilizar

Se realiza una revisión del estado del arte, en donde se estudiaron trabajos anteriormente realizados con metodologías similares, para poder tener una base de inicio en la elección del método

de aprendizaje máquina y de las métricas de evaluación a tener en cuenta.

7.3. Etapa III: Obtención del conjunto de datos

Se obtiene un conjunto de datos el cual contiene el numero de casos necesarios y suficientes para entrenar la red neuronal y lograr la identificación de la inversa de la planta. La toma de datos se realiza sobre el rango de trabajo identificado anteriormente.

7.4. Etapa IV: Entrenamiento de la red neuronal

Después de elegida la arquitectura de la red neuronal, el algoritmo de aprendizaje y las métricas de evaluación, se realizan los entrenamientos y las pruebas sobre la red neuronal seleccionada, al igual que la observación de sus respectivas métricas en cada entrenamiento.

7.5. Etapa V: Implementación del controlador diseñado

Luego de entrenar la red neuronal se implementa como controlador inverso en la plataforma simulada del péndulo invertido. En esta etapa se realizan varias pruebas con las diferentes arquitecturas propuestas al momento de entrenar.

7.6. Etapa VI: Validación de las métricas seleccionadas

Se validan las métricas de evaluación seleccionadas posteriormente para evaluar el desempeño de la red neuronal. Según lo observado en esta etapa, se realizan las correcciones o modificaciones pertinentes sobre la arquitectura de la red neuronal a fin de lograr obtener un mejor desempeño en el controlador neuronal.

7.7. Etapa VII: Comparación del controlador neuronal diseñado con una técnica de control diferente

Se realiza la comparación del controlador neuronal diseñado con respecto a un control LQG implementado sobre la misma plataforma simulada del péndulo invertido. Aquí se muestran las diferencias presentadas entre un controlador con el otro según las métricas seleccionadas y el desempeño de cada uno ante perturbaciones externas.

7.8. Etapa VIII: Análisis y resultados

Se muestran los resultados y análisis de todos los experimentos y pruebas realizadas, desde la toma de datos hasta los diferentes entrenamientos que se hicieron en la red neuronal para validar el control del péndulo.

Estas etapas serán explicadas más detalladamente en los capítulos siguientes. Sin embargo, es importante volver a mencionar que la realimentación vista desde la ETAPA VI a la ETAPA III hace referencia a todas las pruebas realizadas en el proyecto, en aras de obtener los mejores resultados posibles y dar solución a los objetivos propuestos.

Capítulo 8

Selección del Método de Aprendizaje y Métricas de Validación

8.1. Selección del método de aprendizaje

En un proceso de aprendizaje máquina, la definición de la técnica utilizada está relacionada con el objetivo deseado. Una vez establecido este objetivo, se asocia la técnica más adecuada y, posteriormente, se enfrenta al reto de seleccionar el algoritmo o conjunto de algoritmos que ofrezcan el mejor resultado.

Para elegir con éxito el algoritmo que mejor se adapta a las necesidades del proyecto, es importante evaluar qué datos y tipos de datos se tienen disponibles, se trabajará con un juego de datos de entrenamiento previamente etiquetados. Por etiquetado se entiende que para cada ocurrencia del juego de datos de entrenamiento se conoce el valor de su atributo objetivo. Esto le permitirá al algoritmo poder "aprender" una función capaz de predecir el atributo objetivo para un juego de datos nuevo lo que se conoce como aprendizaje supervisado, para este caso se tratará de una regresión ya que el resultado a predecir es un atributo numérico.

8.1.1. Control inverso directo

Se decidió utilizar un control basado en redes neuronales ya que están diseñadas especialmente para reconocer patrones partiendo de la base que esta es capaz de representar cualquier conjunto de pares de entradas y salidas (lineal o no linealmente separables). La aplicación de redes

neuronales sobre sistemas no lineales podrían dar resultados más precisos a la hora de controlar o predecir su comportamiento [23]. Esta identificación se basará en la generación del modelo inverso de la planta, a esto se le conoce como control inverso directo.

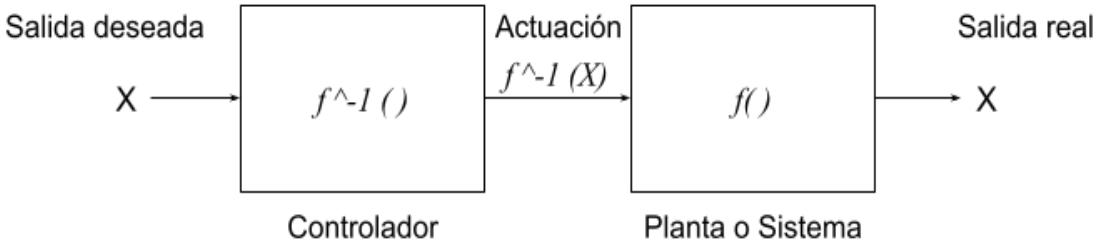

FIGURA 10: planta inversa como controlador [44].

Como se presenta en la figura 10, si se conoce completamente los parámetros de la planta f() se puede generar un controlador $f^{-1}()$ y así obtener la salida deseada que se manifiesta según la identidad $f(f^{-1}(X)) = X$.

Como se quiere generar el modelo inverso de la planta, las entradas de la planta serán las salidas deseadas para la red y las salidas de la planta con dichas entradas corresponderá a la entrada de la red como se muestra en figura 11.

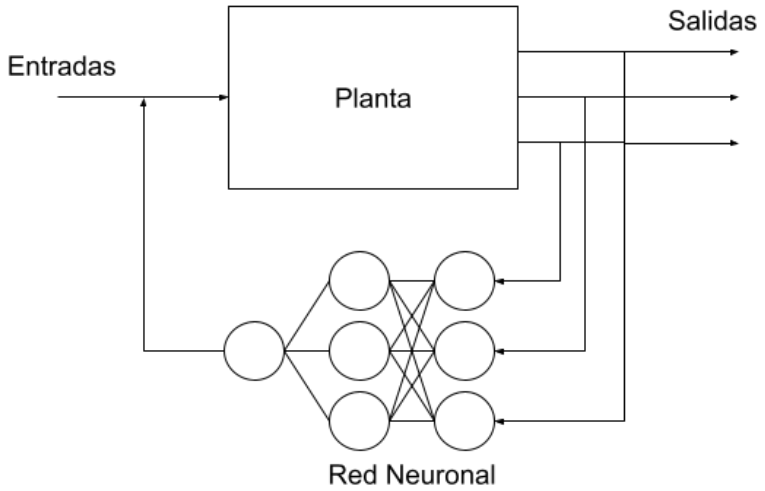

FIGURA 11: modelo de la planta inversa.
[44]

8.1.2. Red neuronal recurrente NARX

Los sistemas dinámicos no lineales pueden ser modelados mediante la red NARX, la cual se ha utilizado en varios estudios. Estos determinan o predicen valores futuros de series temporales a partir de los valores obtenidos, por lo que mencionan que es un modelo que se adapta de la mejor manera a los sistemas dinámicos complejos [26],[45], [46]. Para sacarle el mejor provecho, a la red neuronal NARX para las predicciones de series de tiempo, es necesario darle el mejor uso a la memoria de la red, con sus valores pasados de las series temporales, las mismas que son estimadas o encontradas, y son usadas en forma de retroalimentación para futuras respuestas [47]. En la figura 12 se observan bloques TD, los cuales representan las unidades de retardo de la señal de entrada como para la señal de salida y estos bloques le permiten a la red NARX, que analice las propiedades del sistema, para los procesos dinámicos a los que se sometió la red en el proceso de entrenamiento.

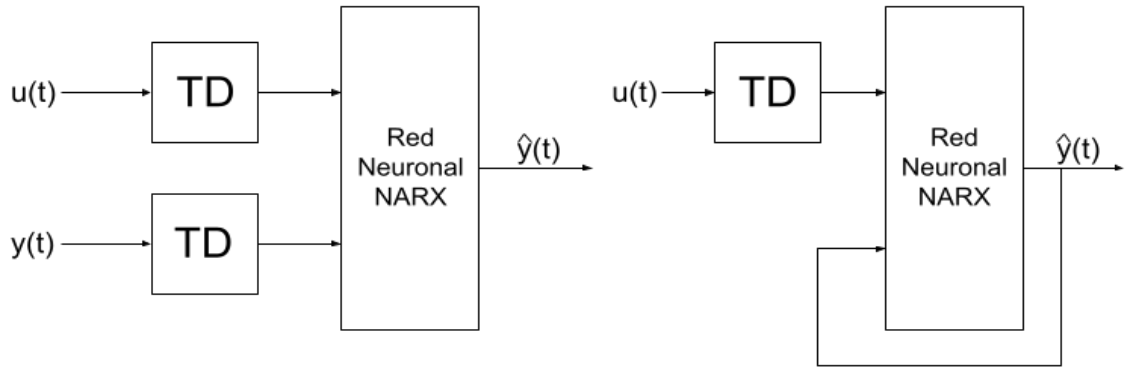

FIGURA 12: Arquitectura de entradas en Serie y Paralelo del modelo NARX [44].

8.1.3. Arquitectura de la red neuronal NARX

Se decidió usar una red neuronal NARX recurrente de alimentación hacia adelante (feedforward) de 3 capas; la capa de entradas, la capa oculta y la capa de salida.

Para la selección del número de neuronas en las capas ocultas y los retardos se hicieron pruebas pertinentes que serán presentadas en la siguiente sección, obteniendo un número adecuado para que el comportamiento sea el esperado, puesto que no existe una regla general para la selección de dichos parámetros [48]. La función de transferencia usada en la capa oculta es sigmoidea. La función de activación utilizada para la capa de salida es lineal, pues no modifica la salida calculada por la capa oculta además de que puede tomar cualquier valor dados datos consistentes y suficientes neuronas en su capa oculta [49]. La red se entrenó con el algoritmo

de retropropagación de Levenberg-Marquardt(LMA) el cual provee una solución numérica al problema de minimización de una función, generalmente no lineal. LMA interpola entre los algoritmos de Gauss-Newton (GNA) y descenso de gradiente donde es considerado mucho más robusto que estos, dado que en la mayoría de los casos encuentra una solución incluso si se inicializa lejos del mínimo. Por otro lado en ciertas funciones y parámetros de inicialización se comporta un poco más lento que el algoritmo GNA [50]. LMA es uno de los más precisos aunque tome mayor esfuerzo computacional comparado con otros algoritmos como Retropropagación gradiente conjugada y el de GNA [51].

Al momento de trabajar con series de tiempo, las redes NARX se presentan como una gran opción por lo mencionado anteriormente. Así como se aprecia en el cuadro 1 tomado de [22] donde compara un sistema de predicción de series temporales usando tres métodos diferentes, como lo fue el método de regresión lineal, de regresión logística y una Red Neuronal Recurrente, evaluando su desempeño en función del error cuadrático medio y del tiempo de inferencia.

Método	**Regresión Lineal**	**Regresión Logística**	**RNN**
MSE	9.0	100.0	2.8
Tiempo	0.37	0.43	3.89

CUADRO 1: Comparación general entre los modelos seleccionados según precisión y velocidad [22].

Como se puede apreciar en la métrica del error cuadrático medio, que es la seleccionada en el artículo para determinar qué método es el más adecuado para el desarrollo de su sistema, la red neuronal recurrente presenta más precisión que los otros dos métodos, sin embargo, también se aprecia que el tiempo de entrenamiento es mayor.

De la misma manera, en el cuadro 2 se puede evidenciar también la comparativa realizada en [26], entre 5 arquitecturas y algoritmos diferentes para un sistema de predicción de lluvia con series temporales. Usando como métricas de evaluación el error cuadrático medio (MSE) y la diferencia porcentual de la raíz media cuadrática (PDR).

Estos datos se obtienen a partir del entrenamiento de cada arquitectura con 4 cuatro conjuntos diferentes. Como se puede apreciar en la métrica del error cuadrático medio, la cuál es la métrica que se usa para medir la evaluación del neurocontrolador desarrollado en el proyecto, las redes neuronales NARX con los algoritmos Levenbergt-Marquardt presentan más precisión que los otros 3 métodos en todos o la mayoría de entrenamientos con los diversos conjuntos de datos, lo cual concluye que esta arquitectura presenta mejor comportamiento y es la seleccionada como mejor alternativa para realizar la predicción.

MÉTRICAS	REGRESIÓN LINEAL	W-SAS-MP	RBF-HPSOGA	LM-NARX	SALM-NARX
PRIMER CONJUNTO DE DATOS					
MSE	0.059	0.513	0.010	0.026	0.008
PRD	9.169 %	15.392 %	2.841 %	5.239 %	1.721 %
SEGUNDO CONJUNTO DE DATOS					
MSE	5.416	2.553	4.495	0.523	0.518
PRD	17.748 %	4.520 %	17.205 %	3.197 %	2.990 %
TERCER CONJUNTO DE DATOS					
MSE	0.063	0.610	0.053	0.028	0.023
PRD	35.762 %	14.670 %	19.770 %	8.694 %	6.492 %
CUARTO CONJUNTO DE DATOS					
MSE	0.036	0.578	0.028	0.027	0.020
PRD	8.794 %	16.158 %	7.816 %	7.666 %	4.834 %

CUADRO 2: Comparación entre diferentes arquitecturas [26].

Tomando el cuadro 1 y el cuadro 2, se observar que en ambos casos el comportamiento de una red neuronal recurrente, como lo es la red neuronal de arquitectura NARX, presenta mejor desempeño con respecto a otras arquitecturas de red neuronal, a partir de métricas de evaluación como el error cuadrático medio. En el cuadro 3 se muestra un resumen de estas tablas que presentan la selección de la red neuronal NARX como la arquitectura a utilizar en el desarrollo de este proyecto.

ARTÍCULO 1					
MÉTRICAS	**REGRESIÓN LINEAL**	**REGRESIÓN LOGÍSTICA**		**RED N. RECURRENTE**	
MSE	9.0 [22]	100.0 [22]		2.8 [22]	
ARTÍCULO 2, ENTRENAMIENTO CON MEJOR RESULTADO					
MÉTRICAS	**REGRESIÓN LINEAL**	**W-SAS-MP**	**RBF-HPSOGA**	**LM-NARX**	**SALM-NARX**
MSE	0.059 [26]	0.513 [26]	0.010 [26]	0.026 [26]	0.008 [26]

CUADRO 3: Resumen de la comparación de arquitecturas (Fuente: Autores).

8.2. Selección de las métricas de evaluación

Para medir el rendimiento de la red se tuvo en cuenta la métrica MSE o Error Cuadrático Medio, el sobrepaso máximo y la respuesta ante perturbaciones, tomando de base el estudio realizado en el estado del arte donde se tomaban estos parámetros para medir el desempeño de los sistemas y se garantizaban un buen resultado.

El cuadro 3 usado anteriormente para la selección de la arquitectura, muestra la evaluación del desempeño comparando diferentes arquitecturas, a partir de la métrica del error cuadrático

medio (MSE), mostrando que entre más cercano sea a 0 este valor, mejor será el desempeño del controlador. Por tal motivo, este cuadro también es usando como sustento para la elección de la métrica MSE en el desarrollo de este proyecto.

Tal como se observa en el cuadro 4 tomado de [52] donde se presentan los valores de error presentados en un sistema de péndulo invertido, usando una Red Neuro-Difusa frente a un control PID. Se menciona que gracias a la implementación de la red, se logra reducir el valor de error, llegando a presentar una mejor estabilidad en el péndulo.

Nombre del Modelo	**Error Obtenido**
Control PID	0.015336
Red Neuro-Difusa	0.01533

CUADRO 4: Evaluación de un sistema de péndulo invertido usando una Red Neuro-Difusa y un control PID [52].

El cuadro 4 muestran diferencias muy mínimas en la obtención del error. El artículo citado opta por implementar los dos controles en la planta, llegando a obtener mejor resultados en el error.

En el cuadro 5 se muestran los resultados de la comparación de dos controles clásicos implementados en el sistema del péndulo invertido, usando como métricas de desempeño el sobrepaso máximo y el tiempo de establecimiento. Lo controles usados fueron; el Control Clásico PID y un control FOPID. Estos datos son tomados de [53].

Métrica	**Control IOPID**	**Control FOPID**
Sobrepaso	$5{,}959 * 10^{-3}$	$4{,}497 * 10^{-3}$
Tiempo de establecimiento	2.056 s	1.761 s

CUADRO 5: Comparación entre un Control IOPID y un Control FOPID aplicados en un péndulo invertido [53].

En [53] muestran la importancia de tener un bajo sobrepaso y un tiempo de establecimiento corto, por lo que se intenta buscar el control que mejor pueda estabilizar el sistema bajo estos parámetros. Para el caso de este proyecto se debe tener en cuenta que el péndulo parte desde una posición ya establecida dentro la región de trabajo, es decir, que este tiempo de establecimiento no se tiene en cuenta ya que desde un momento inicial este estaría establecido en el punto a controlar.

También se visualizan los resultados obtenidos en [54], donde muestra la implementación de un control LQG comparado con controles robustos. Aquí muestran la importancia de tener los tres métricas de evaluación lo mejor establecidas posibles. Los resultados se pueden observar en el cuadro 6.

Métricas	Control LQG	Control H-infinity Loop-shaping
Tiempo de estabilización	1.51 s	2.43 s
sobrepaso máximo	0.0124	0.023s
Error	0.015	0.05

CUADRO 6: Comparación entre un Control LQG y un Control H-infinity Loop-Shaping aplicados en un péndulo invertido. [54].

De lo anterior, se puede observar que el controlador con mejor rendimiento es el LQG, ya que todas sus métricas así lo reflejan, haciendo que la precisión del control sea más alta. Eso mismo es lo que se espera en el desarrollo de esta práctica.

En cuadro 7 muestra el resumen de los cuadros anteriormente mostrados, donde se hace una comparativa entre estas y se analiza el desempeño de cada arquitectura con respecto a la métrica seleccionada. En cada trabajo se concluye que aquella arquitectura que presente el mejor error cuadrático medio (MSE) será seleccionada como la mejor alternativa de control.

ARTÍCULO 1			
MÉTRICAS	**REGRESIÓN LINEAL**	**REGRESIÓN LOGÍSTICA**	**RED N. RECURRENTE**
MSE	9.0 [22]	100.0 [22]	2.8 [22]

ARTÍCULO 2, ENTRENAMIENTO CON MEJOR RESULTADO					
MÉTRICAS	**REGRESIÓN LINEAL**	**W-SAS-MP**	**RBF-HPSOGA**	**LM-NARX**	**SALM-NARX**
MSE	0.059 [26]	0.513 [26]	0.010 [26]	0.026 [26]	0.008 [26]

ARTÍCULO 3		
MÉTRICAS	**CONTROL PID**	**RED NEURO-DIFUSA**
MSE	0.015336 [52]	0.01533 [52]

ARTÍCULO 4		
MÉTRICAS	**CONTROL IOPID**	**CONTROL FOPID**
SOBREPASO MÁXIMO	$5{,}959 * 10^{-3}$ [53]	$4{,}497 * 10^{-3}$ [53]
TIEMPO DE ESTABILIZACIÓN	2.056 Segundos [53]	1.761 Segundos [53]

ARTÍCULO 5		
MÉTRICAS	**CONTROL LQG**	**CONTROL H-INFINITY LOOP-SHAPING**
MSE	0.015 [54]	0.05 [54]
TIEMPO DE ESTABILIZACIÓN	1.51 Segundos [54]	2.43 Segundos [54]
SOBREPASO MÁXIMO	0.0124 [54]	0.023 [54]

CUADRO 7: Resumen de la comparación de Métricas (Fuente: Autores).

Con las métricas establecidas se dispone a realizar la toma de datos, luego el entrenamiento de la red, la implementación completa con la planta y la validación de estas métricas visto frente a un control LQG implementado en la misma plataforma.

Capítulo 9

Análisis y Resultados

En este capitulo se mostrará los resultados de las diferentes pruebas y experimentos que se realizaron para dar cumplimiento a los objetivos teniendo en cuenta el diseño metodologico planteado, el método de aprendizaje seleccionado y la métricas que se tendrán en cuenta para evaluar el desempeño del controlador.

9.1. Conjunto de datos para el entrenamiento de la red neuronal

Es necesario recolectar los datos que describen correctamente el comportamiento real del sistema, pues de esto depende el desarrollo satisfactorio de las etapas siguientes y en consecuencia un buen control para el sistema [55]

9.1.1. Entradas y salidas de la red neuronal

Los datos de entrada son el ángulo del péndulo (alpha), el ángulo del brazo giratorio (tetha), el error (la diferencia entre la referencia y el ángulo alpha) y el voltaje inducido a la planta (Vm) con los respectivos retardos a cada entrada. La salida del sistema es Vm.

9.1.2. Rango de trabajo

El control solo trabajará en un rango de alpha ya que se demostró con los controladores existentes como el LQG o el PID que están diseñados para trabajar en un rango específico que es de

$154°$ a $206°$, teniendo en cuenta que en $180°$ (π) es cuando el péndulo se encuentra en posición vertical como se muestra en la figura 13 , es decir, el punto de trabajo es de $\pm 26°$. Tal como se observa en la figura 14.

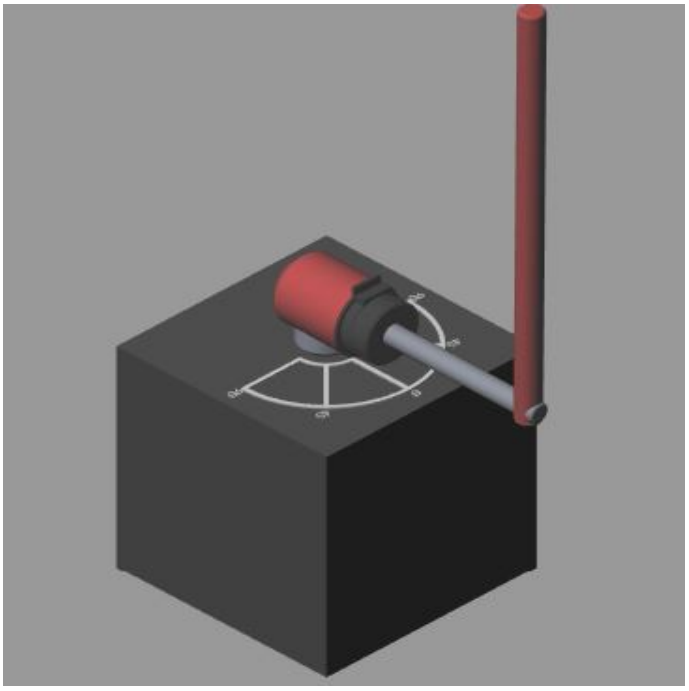

FIGURA 13: Alpha en posición vertical ($180°$) (Fuente: Autores).

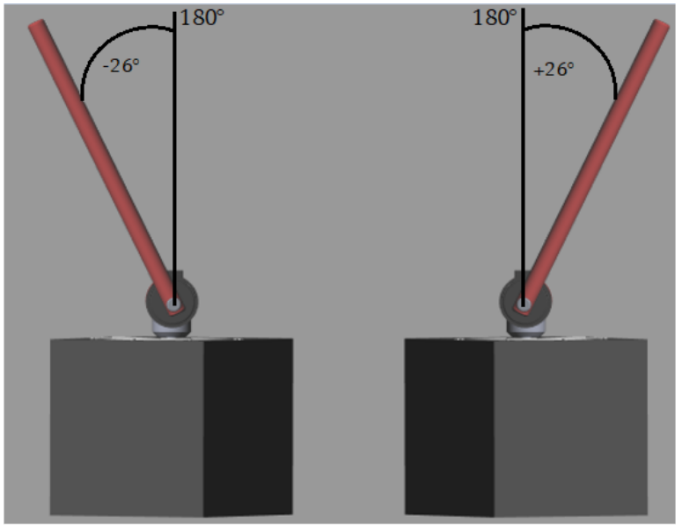

FIGURA 14: Rango de trabajo de Alpha (Fuente: Autores).

9.1.3. Recolección de datos

Se aplicará una señal sinusoidal a la entrada del péndulo (Vm) con el fin de identificar el comportamiento de las salidas (theta, alpha y el error). Inicialmente la toma de datos se propuso como se muestra en la figura 15. Para el voltaje (Vm) se usó una señal sinusoidal con la frecuencia y amplitud necesaria para llevar el péndulo al punto de trabajo, se encontró que con una señal de amplitud de 22V y una frecuencia de 12Hz se cumplía esta acción.

Frecuencia de muestreo

En esta parte es fundamental tener en cuenta el teorema de Nyquist que como se explica anteriormente afirma que la velocidad de muestreo Fs debe ser mayor que el doble del componente de interés de frecuencia más alto en la señal medida Fn.

$Fs > 2Fn$
$Fs > 24Hz$

Aunque la frecuencia de la señal de la salida de alpha y theta se desconocen, además no se puede calcular analizando la gráfica ya que no se trata de una señal periódica. En la simulación que viene por defecto con el bloque del péndulo invertido se maneja un tamaño de paso automático, es decir, que Matlab asigna ese valor y puede ser variante con el tiempo de simulación, lo que viene siendo un problema ya que para la implementación de la red descrita anteriormente es fundamental saber el tiempo de muestreo (Ts), adicionalmente se necesita que este sea fijo para aplicar correctamente los retardos en la capa de entradas de la red neuronal. Para esto se corrió la simulación por defecto (con Ts automático) por 3 segundos y se evidenció en el workspace que en ese tiempo había tomado 59336 datos, aproximadamente un tiempo de muestreo de 20 milisegundos así que se asigno este valor fijo a la hora de la toma de datos.

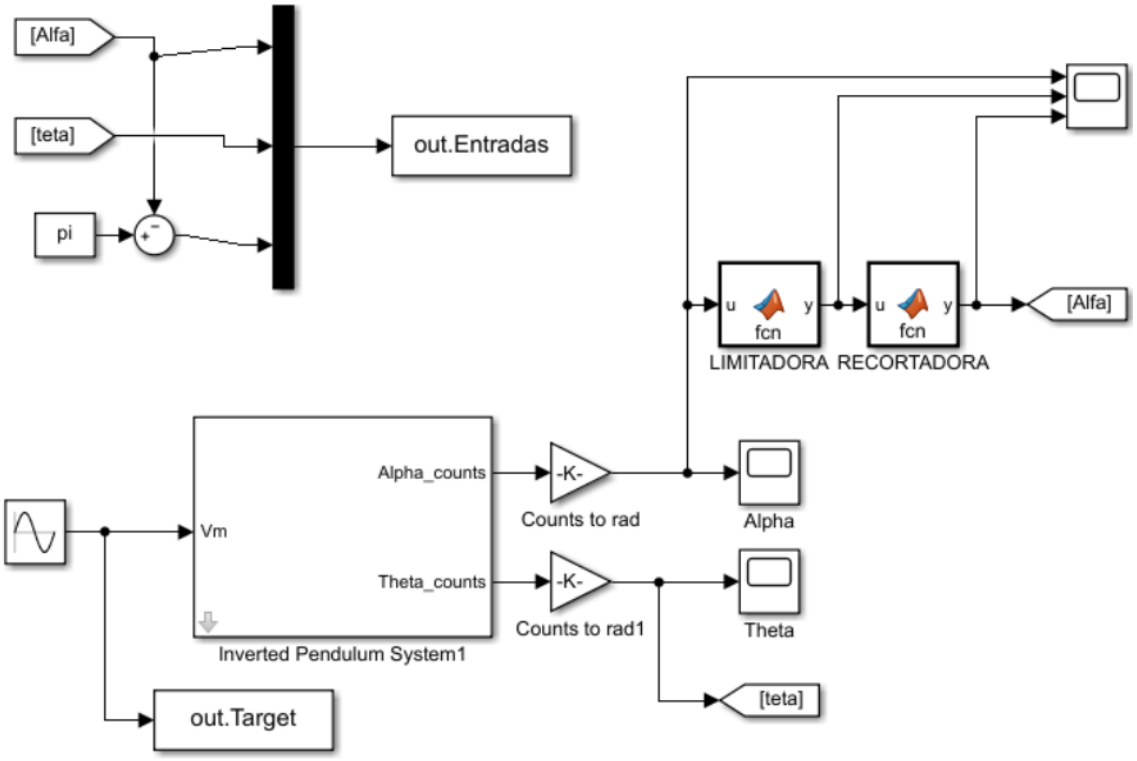

FIGURA 15: Adquisición de datos para la identificación del modelo (Fuente: Autores).

Con este método de adquisición de datos se realizaron varios entrenamientos cambiando el numero de neuronas de la capa oculta y el algoritmo de aprendizaje pero aun así el error era demasiado alto y al momento de implementar el controlador no funcionaba como se muestra mas adelante en esta sección (figura 27), donde se entrenará la red modificando el conjunto de datos de entrada, la arquitectura y las entradas.

Luego de realizar las pruebas pertinentes para comprobar el funcionamiento del neurocontrolador se infirió que los datos no estaban brindando la suficiente información para lograr el objetivo de mantener el péndulo en el rango de trabajo, así que se decidió aplicar dos señales sinusoidales a Vm como se muestra en la figura 16; una se encarga de llevar el péndulo hasta el punto de trabajo yla otra señal para que el péndulo permaneciera por más tiempo en dicho rango con una frecuencia más alta y una amplitud más baja como se muestra en la figura 17. Se observa que estas dos señales están conectadas a una función que también tiene como entrada el ángulo de alpha con el fin de que pueda identificar el momento en el que el péndulo se encuentra en el punto de trabajo y así variar Vm.

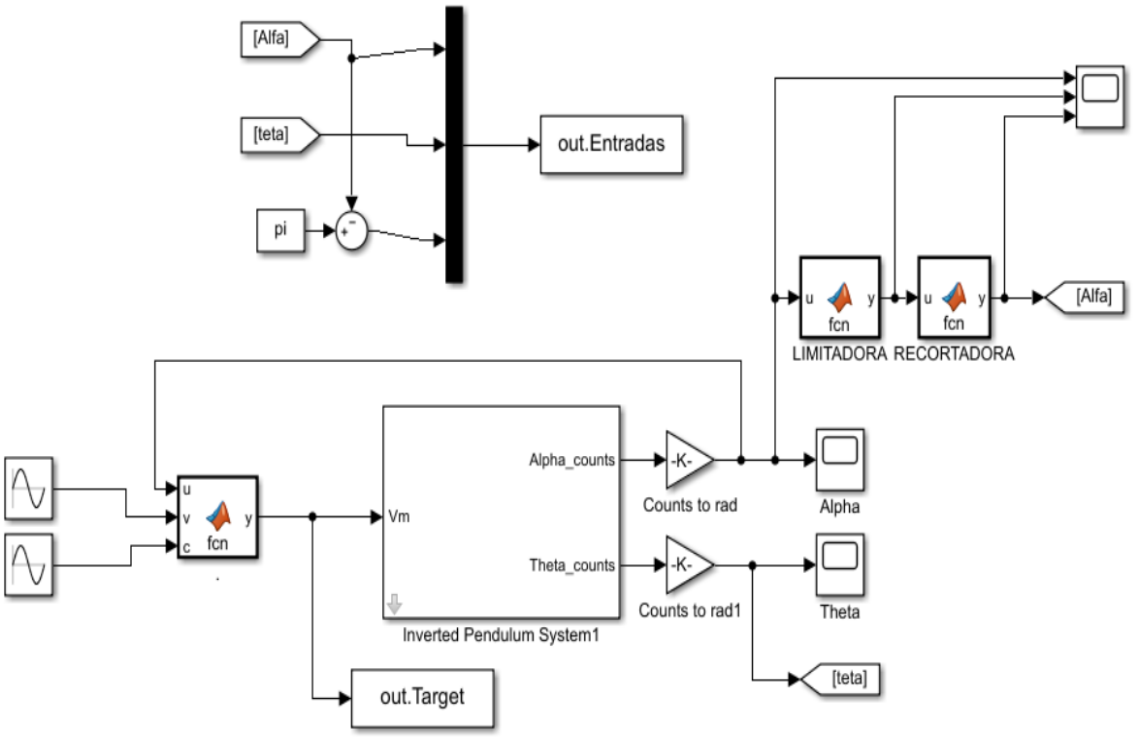

FIGURA 16: Adquisición de datos para la identificación del modelo con dos entradas (Fuente: Autores).

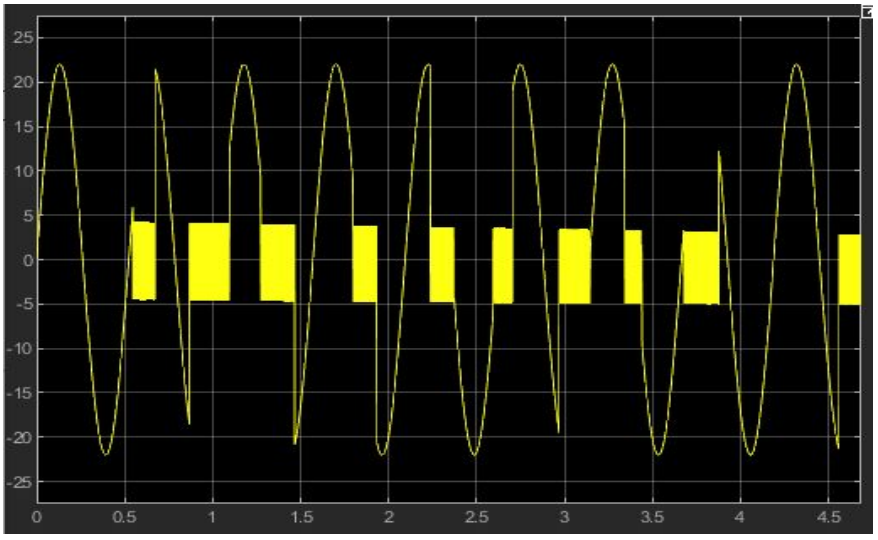

FIGURA 17: Voltaje inducido a la planta en la toma de datos (Fuente: Autores).

9.1.3.1. Preparación de los datos

Las señales de salida del ángulo alpha y theta se les convierten los grados a radianes, así que cuando el péndulo se encuentre equilibrado se esperaría que el valor de alpha tenga que ser $180°$ o π, pero no siempre es así ya que los giros completos de $360°$ (donde el signo depende del sentido del giro) se acumulan y esto cambia la referencia. Esto se logra entender a partir de la figura 18, donde se observa como el valor del ángulo alpha empieza a aumentar ya que el voltaje inducido al péndulo por medio de las señales de entrada hacen que este solo gire hacia la derecha aumentando los grados continuamente.

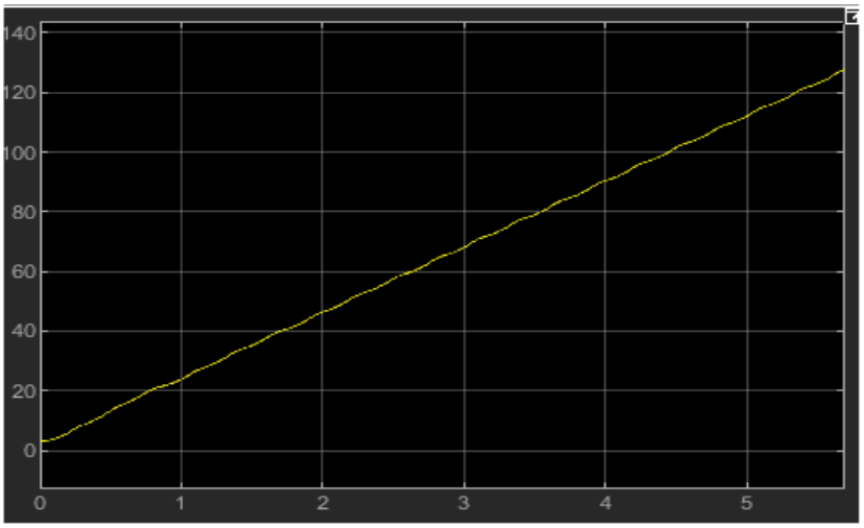

FIGURA 18: Ángulo alpha cuando el péndulo está girando hacia la derecha (Fuente: Autores).

Para corregir este problema, se crea una función llamada "LIMITADORA", vista en la figura 16 que tiene como objetivo mantener los valores de alpha entre -2π a 2π, es decir, cuando la función detecta que el péndulo da un giro, sea a la derecha o izquierda, convierte alpha en 0 para que de esta manera no se empiece a acumular este valor y siempre que se encuentre en el punto de equilibrio el valor de alpha equivaldrá a π. La respuesta de esta función se refleja en la figura 19.

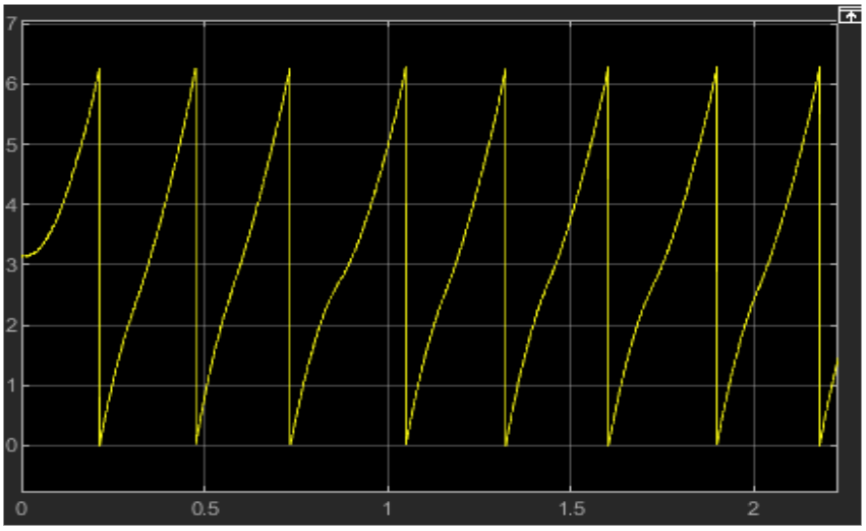

FIGURA 19: Valor de alpha cuando el péndulo está girando a la derecha con la función limitadora (Fuente: Autores).

Una vez obtenido los valores del valor del ángulo alpha en este rango de -2π a 2π, se crea la función "RECORTADORA", que se encarga de poner en 0 los valores de alpha que están fuera del rango de trabajo, tal como se aprecia en la figura 20. Estos datos serán almacenados por los bloques 'to File' en la carpeta archivos donde posteriormente se creará un arreglo mediante un archivo .m que se encargue de eliminar todos los datos que no estén dentro del rango para luego ser almacenados en el workspace.

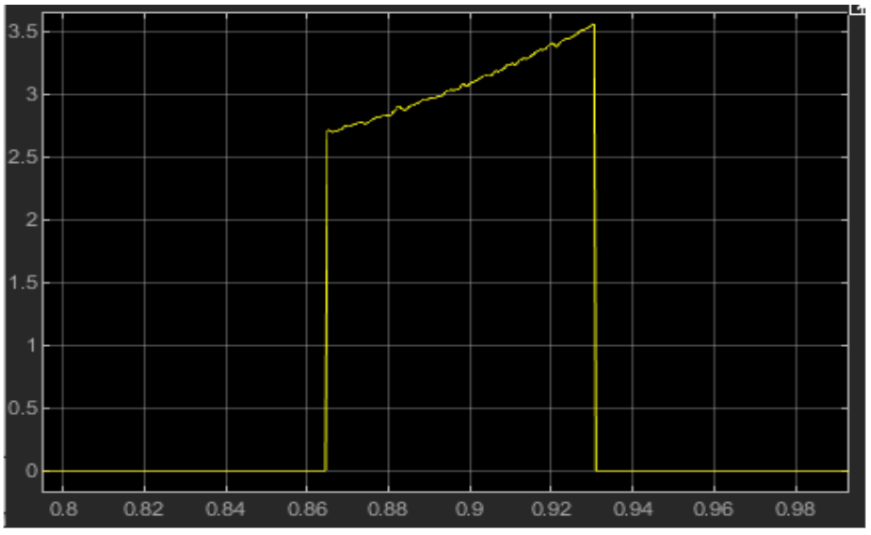

FIGURA 20: Valor de alpha con la función recortadora (Fuente: Autores).

9.2. Entrenamiento de la red neuronal e implementación del controlador.

Luego de tener el conjunto de datos, se realiza el entrenamiento con ayuda de la aplicación neural network time series app de Neural Network toolbox la cual permite:

1. Seleccionar arquitectura

La aplicación neural network time series app de Neural Network toolbox, la cual es muy útil al momento de trabajar con predicciones de tiempo, brinda 3 opciones a la hora de trabajar con esta clase de problemas tal como se muestra en la figura 21.

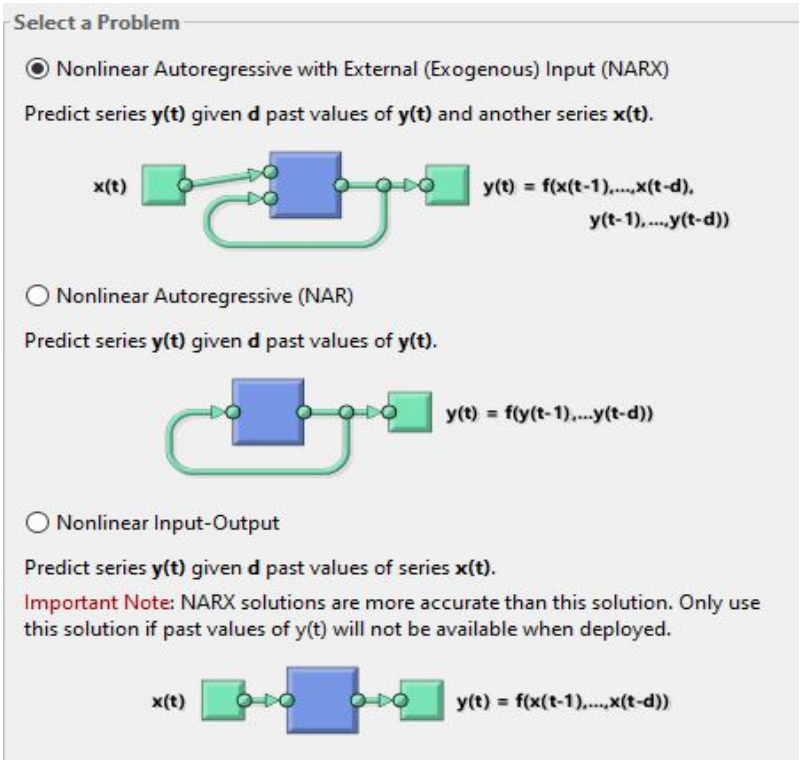

FIGURA 21: Opciones de neural network time series app (Fuente: Autores).

Se elige la arquitectura NARX por las razones mencionadas en el capítulo de "Selección de métricas de aprendizaje y métricas de evaluación".

2. Seleccionar conjunto de datos a entrenar

Luego de seleccionar la arquitectura, se debe indicar las entradas y salidas de la red con la cual sera entrenada como se muestra en la figura 22.

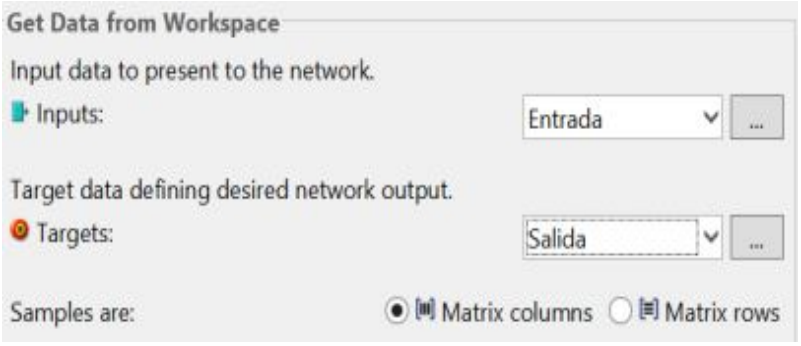

FIGURA 22: Selección de los datos de entrada y salida de la red (Fuente: Autores).

Luego se dividen los datos en conjuntos de entrenamiento, validación y prueba, como se muestra en la figura 23.

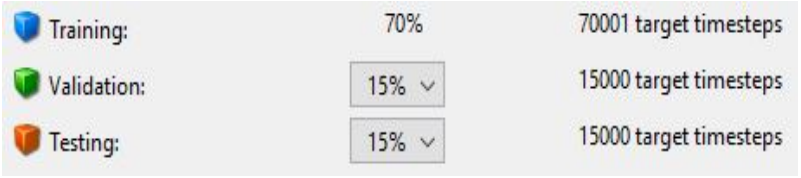

FIGURA 23: Selección de conjuntos de entrenamiento, validación y prueba (Fuente: Autores).

- 70 % Para entrenamiento, es el que se usa para calcular el gradiente y actualizar los pesos y bías de la red.
- 15 % Para validar que la red se está generalizando y dejar de entrenar antes de sobreajustar.
- 15 % Para probar de forma independiente la generalización de la red.

3. Seleccionar parámetros de la red y entrenamiento

Se selecciona el numero de neuronas que se quiere en la capa oculta y el numero de retardos a la entrada como se muestra en la figura 24

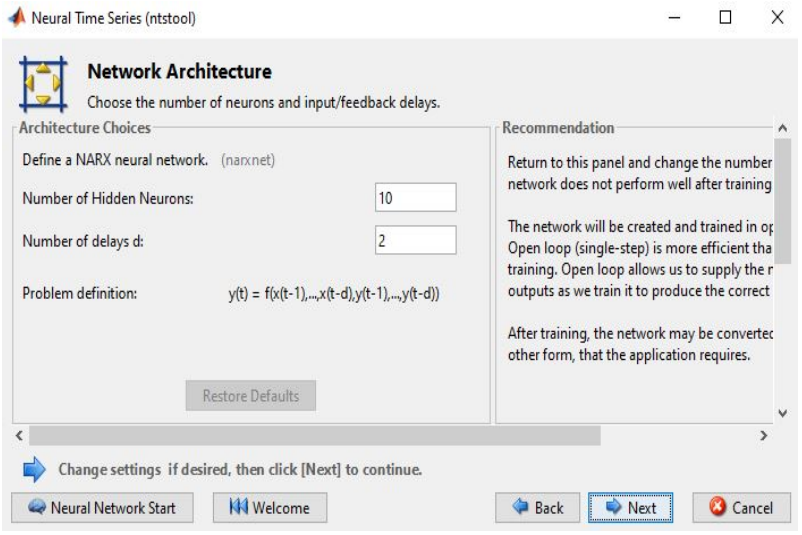

FIGURA 24: Selección de parámetros de la red (Fuente: Autores).

Como se mencionó en el capítulo anterior se usa el algoritmo de Levenberg-Marquardt, En el panel Entrenamiento de la figura 25 se puede ver el progreso del entrenamiento y los criterios de detención, el entrenamiento continúa hasta que se cumple uno de los criterios.

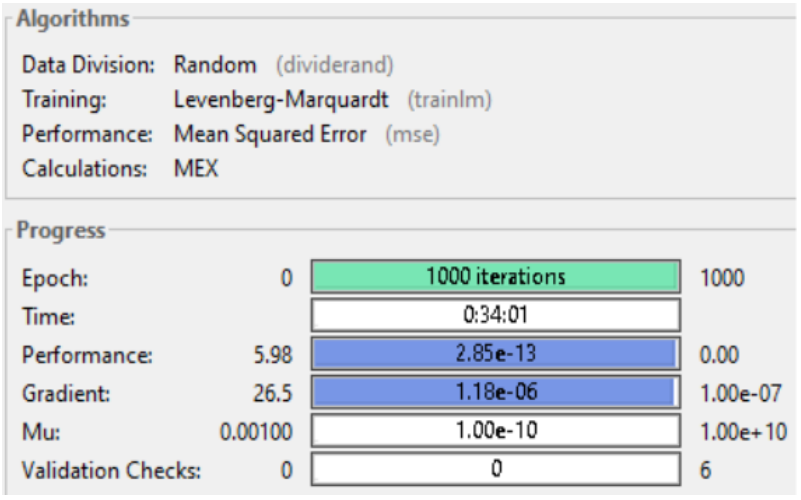

FIGURA 25: Panel de entrenamiento y criterios de detención (Fuente: Autores).

4. Implementar la red Neuronal en un bloque de Simulink y evaluación de las métricas eleccionadas

Luego del entrenamiento la aplicación facilita la obtención de una función de la red, el script y un modelo el simulink el cual es usado al momento de implementar la red neuronal como controlador inverso de la planta, tal cómo se muestra en la figura 26, teniendo en cuenta que a

la entrada del ángulo alpha se le pasará la constante π directamente ya que es el valor deseado. El ángulo inicial de alpha es de $180°$.

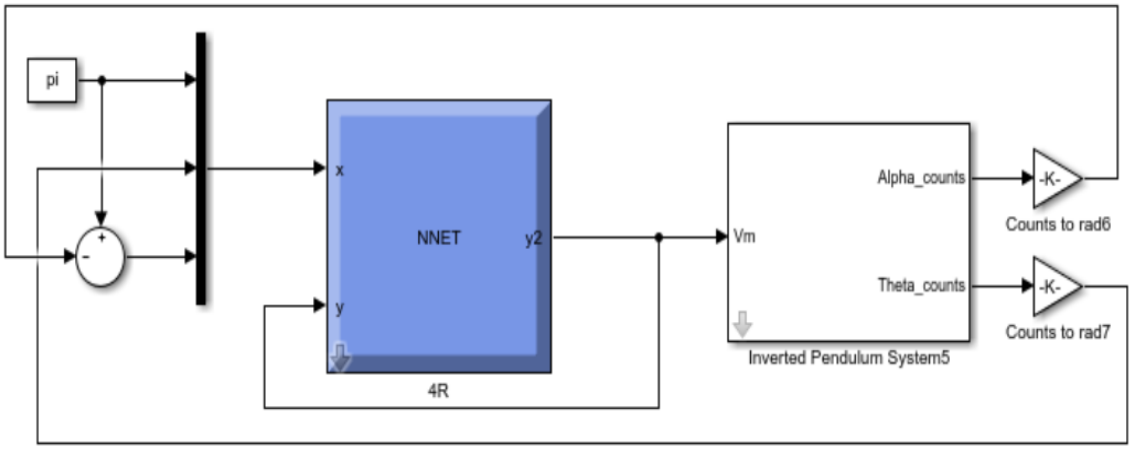

FIGURA 26: Implementación del controlador con red neuronal inversa (fuente: Autores).

Inicialmente se había planteado la toma de datos como se había descrito anteriormente, con una sola señal sinusoidal a la entrada que llevara el péndulo al punto de trabajo pero por mas que cambiaran los parámetros de entrenamiento con esos conjuntos de datos, el error era demasiado alto y al momento de implementar el controlador no lograba su objetivo. El menor MSE fue de 2.506 como se muestra en la figura 15 lo cual es un error muy alto. Finalizado el entrenamiento se implementó el controlador pero como era des esperarse no logro mantener el péndulo en la posición vertical (figura 28), el proceso se realizo varias veces con diferentes conjuntos de datos extraídos de la misma manera y variando los parámetros de la red neuronal pero el MSE no bajaba de 2 sin lograr el control del ángulo.

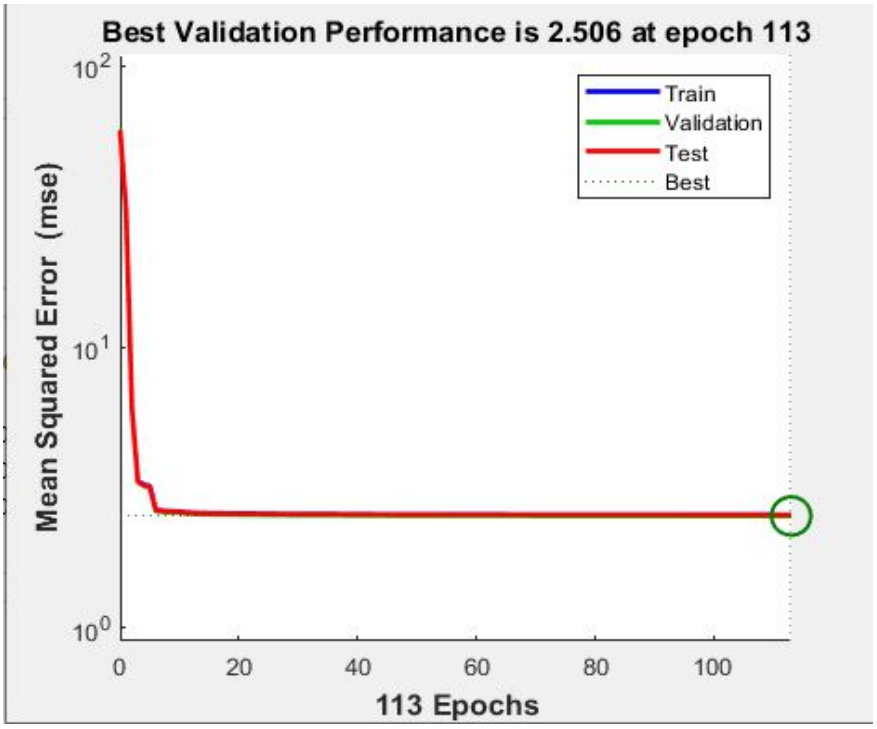

FIGURA 27: MSE del entrenamiento (Fuente: Autores).

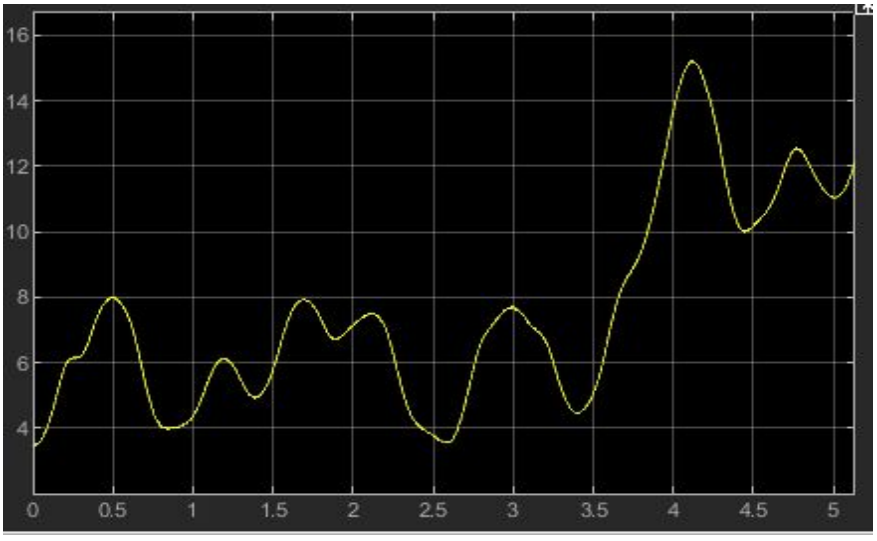

FIGURA 28: Alpha del neurocontrolador (Fuente: Autores).

Luego de realizar las pruebas pertinentes para comprobar el funcionamiento del neurocontrolador se infirió que los datos no estaban brindando la suficiente información para lograr el objetivo de mantener el péndulo en el rango de trabajo así que se decidió aplicar dos señales sinusoidales a Vm como se muestra en la figura 16; una se encarga de llevar el péndulo hasta

el punto de trabajo y otra señal para que el péndulo permaneciera por más tiempo en dicho rango con una frecuencia más alta y una amplitud más baja como se muestra en la figura 17. Se observa que estas dos señales están conectadas a una función que también tiene como entrada el ángulo de alpha con el fin de que pueda identificar el momento en el que el péndulo se encuentra en el punto de trabajo y así variar Vm.

Como se había mencionado anteriormente no existe una norma especifica para la selección de los parámetros de la red así que con el método anteriormente mencionado de obtención de datos se realizaron distintas pruebas variando la arquitectura de la red y sus entradas.

A continuación se explicara el proceso de entrenamiento con las diferentes arquitecturas propuestas, se empieza explicando con la que mejor se tuvo resultados con fines didácticos para lograr realizar una comparación respecto a las redes que no se tuvieron los resultados esperados.

9.2.1. Entrenamiento con 10 neuronas y 4 retardos

Se entrenó la red con la arquitectura que se muestra en la figura 29, con 10 neuronas en la capa oculta que es el valor que viene por defecto y con 4 retardos en las entradas.

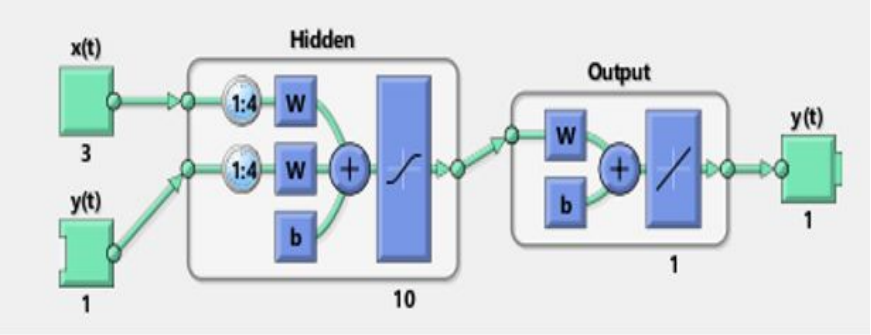

FIGURA 29: Arquitectura con 10 neuronas y 4 retardos (Fuente: Autores).

En este caso cumple con el límite de iteraciones que es de 1000 como se muestra en la figura 25 y llegando a un error de 7.2 x 10-9, tal como se aprecia en la figura 30, este entrenamiento tardo 34 minutos.

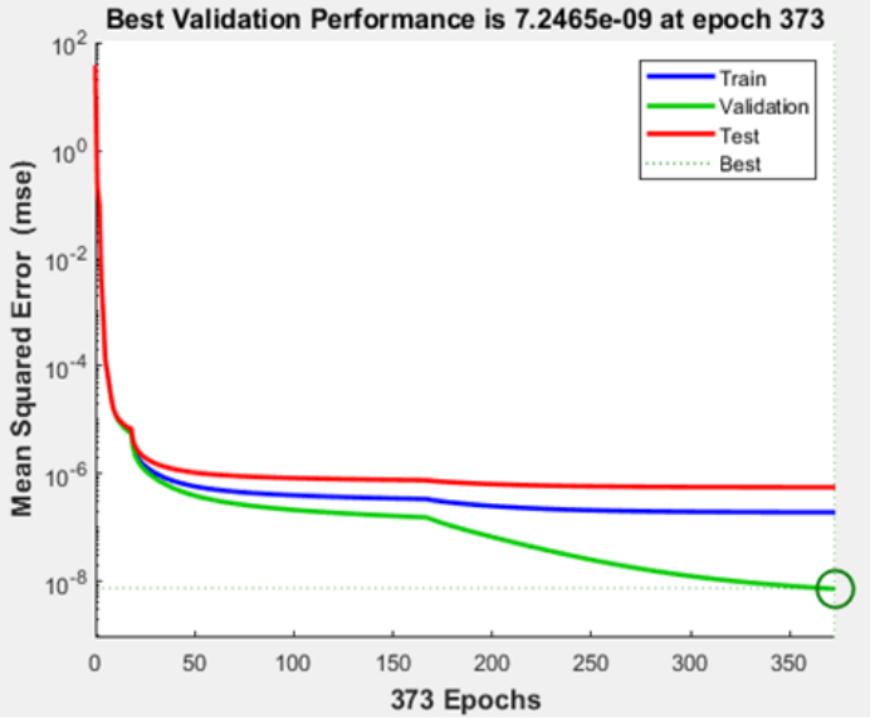

FIGURA 30: MSE del entrenamiento con 4 retardos y 10 neuronas en la capa oculta (Fuente: Autores).

Se procedió a implementar el controlador en la simulación y realizar la prueba del controlador con un minuto y logró mantener el ángulo de alpha dentro del rango de trabajo, como se aprecia en la figura 31.

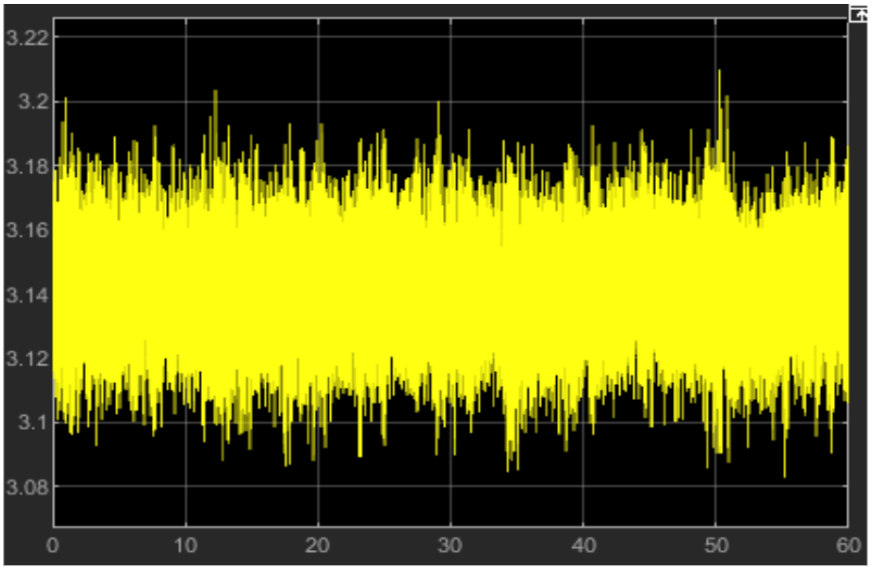

FIGURA 31: Funcionamiento del neurocontrolador (un minuto) (fuente: Autores).

En la figura 32 se visualiza el error, como la señal del error varía tanto en el tiempo se decidió promediar este valor con la función mean(arg) de Matlab donde arg es el arreglo donde se almacenó el error. El neurocontrolador tuvo un error promedio de 0.0239 y un sobrepaso máximo de 0.0917 en el segundo 29.175.

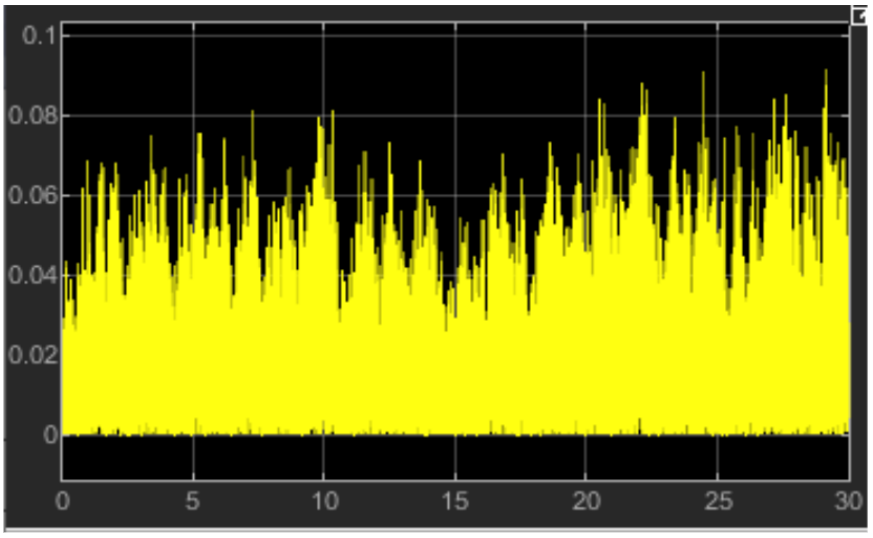

FIGURA 32: Error de alpha respecto a pi del control (fuente: Autores).

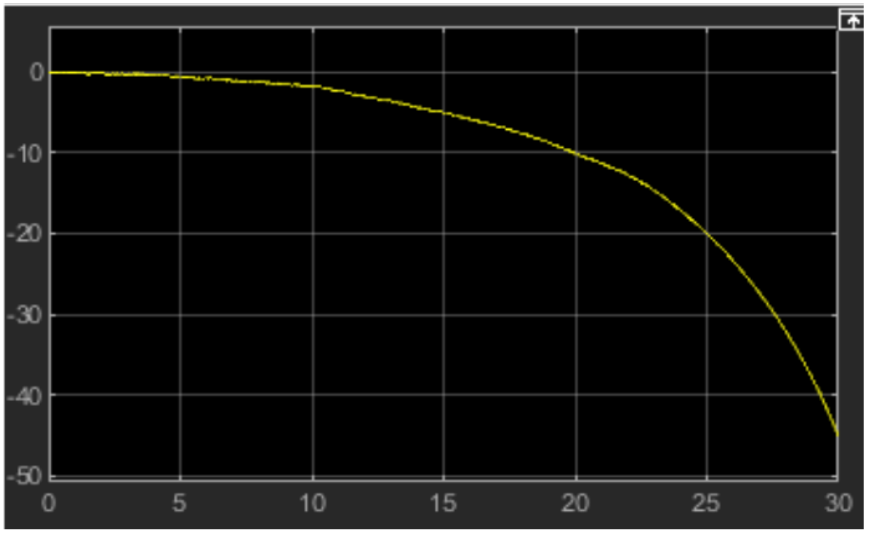

FIGURA 33: Theta neurocontrolador (fuente: Autores).

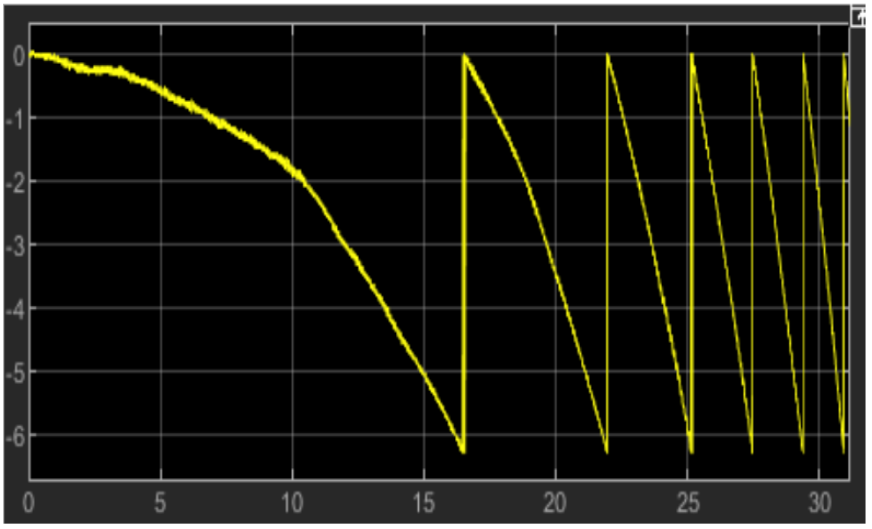

FIGURA 34: Theta neurocontrolador con función limitadora (fuente: Autores).

El la figura 33 se observa que solo da giros en theta en contra de las manecillas del reloj puesto que va de 0 y tiende hacia infinito negativo, se aplica la función limitadora a la salida del ángulo de theta con el fin de que no se acumulen los radianes al dar un giro completo, como se hizo en el proceso de la toma de datos para alpha y se observa que solo da giros en theta en contra de las manecillas del reloj puesto que el valor va de 0 a -2π como se muestra en la figura 34. Se aprecia como aumenta la velocidad a medida que da un giro sobre el ángulo theta (Se sabe que es un giro porque inicia desde 0 hasta -2π), en el primero se demora 16.4s, en el segundo 5.4s, en el tercero 3.2s, en el cuarto 2.3s y así sucesivamente hasta que da el giro tan rápido que se desestabiliza. El péndulo con el control neuronal da aproximadamente 6 giros en theta (360°) y la velocidad con la que aumenta o disminuye dicho ángulo empieza aumentar con el tiempo haciendo perder el control.

Con el mismo set de datos se realizó el entrenamiento aumentando el numero de neuronas pero no se vio mejoría en el MSE y si un aumento de tiempo notable en el entrenamiento, con 6 neuronas demoró aproximadamente 55 minutos y con 4 tardo 44 minutos, así que en lugar de aumentar el número de neuronas en la capa oculta se decidieron disminuir para así evaluar el desempeño.

9.2.2. Entrenamiento con 10 neuronas y 2 retardos

Con el mismo conjunto de datos que se realizó el anterior entrenamiento se ajusto el parámetro de retardos a 2 como se muestra en la figura 35.

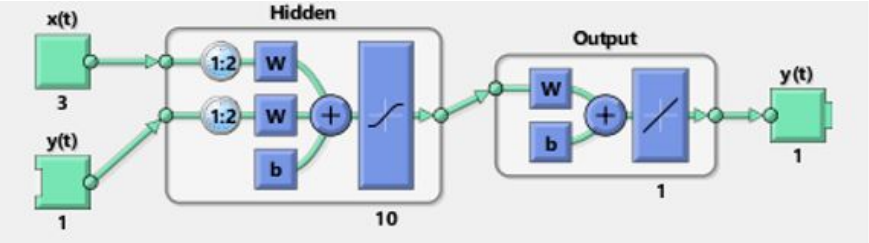

FIGURA 35: Arquitectura con 10 neuronas y 2 retardos (fuente: Autores).

En la figura 36 se evidencia la evolución de MSE convergiendo a un valor de $5{,}39 * 10^{-8}$ finalizando el entrenamiento en la época 679.

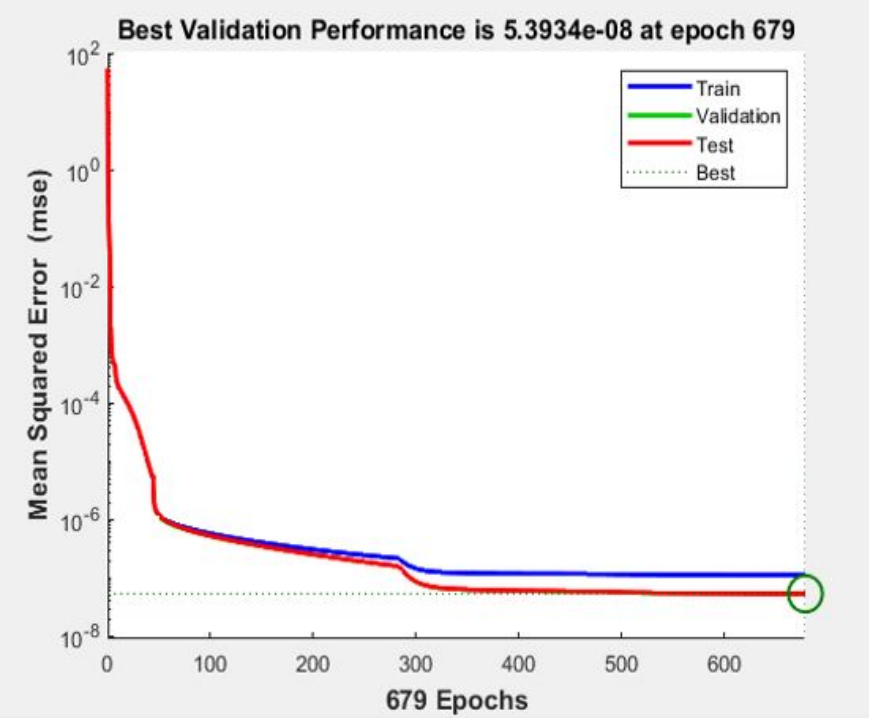

FIGURA 36: Entrenamiento con dos retardos (Fuente: Autores).

A pesar de haber obtenido un MSE muy cercano a 0, cuando se implemento el controlador no logro mantener el ángulo alpha del péndulo en el rango de trabajo por mas de 2 segundos como se muestra en la figura 37.

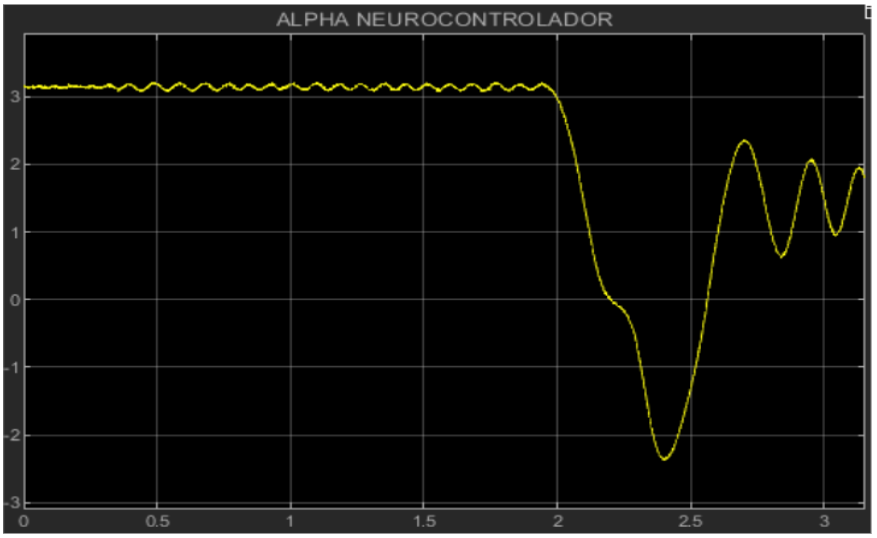

FIGURA 37: Alpha del neurocontrolador con dos retardos (Fuente: Autores).

9.2.3. Entrenamiento con 10 neuronas y 3 retardos

Con el mismo conjunto de datos que se realizó el anterior entrenamiento se ajusto el parámetro de retardos a 3 como se muestra en la figura 38.

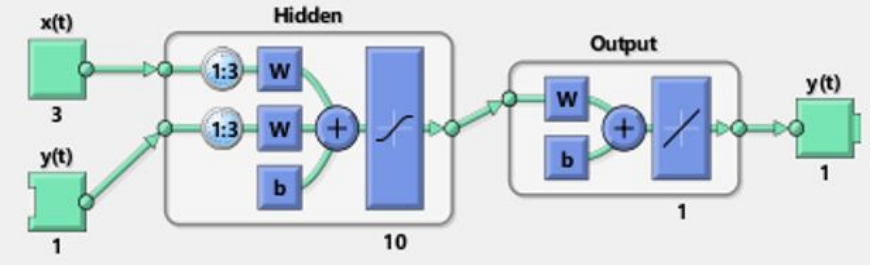

FIGURA 38: Arquitectura con 10 neuronas y 3 retardos (fuente: Autores).

En la figura 39 se evidencia la evolución de MSE convergiendo a un valor de $3{,}23 * 10^{-8}$ finalizando el entrenamiento en la época 199. Un valor muy cercano cuando se realizo este proceso con 2 retardos.

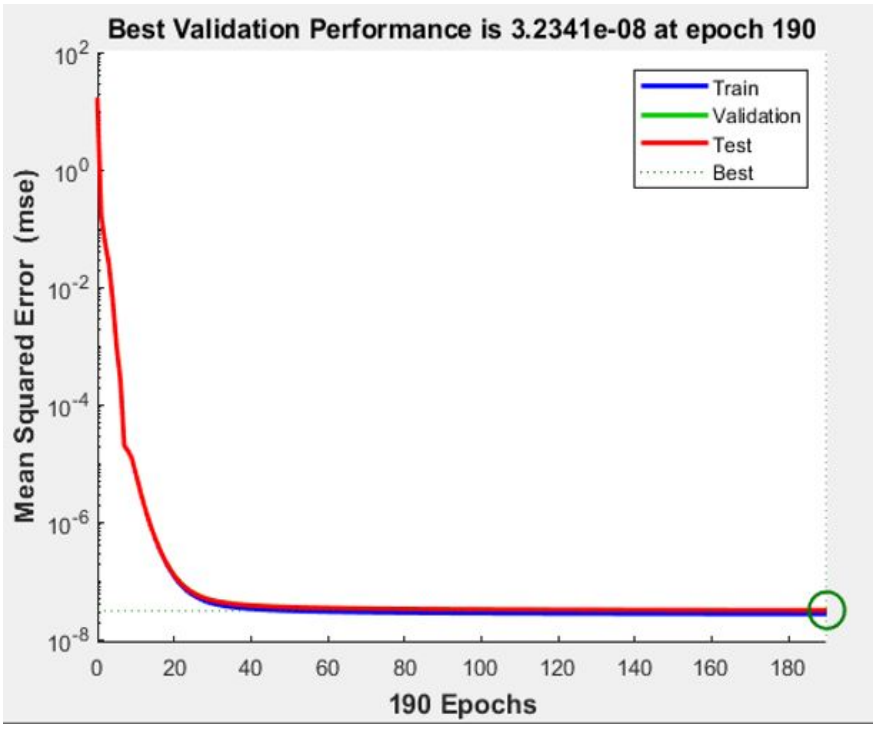

FIGURA 39: Entrenamiento con tres retardos (Fuente: Autores).

Posteriormente se implementó la red entrenada como controlador, duro aproximadamente 20 segundos en el punto de trabajo de alpha del péndulo como se muestra en la figura 40.

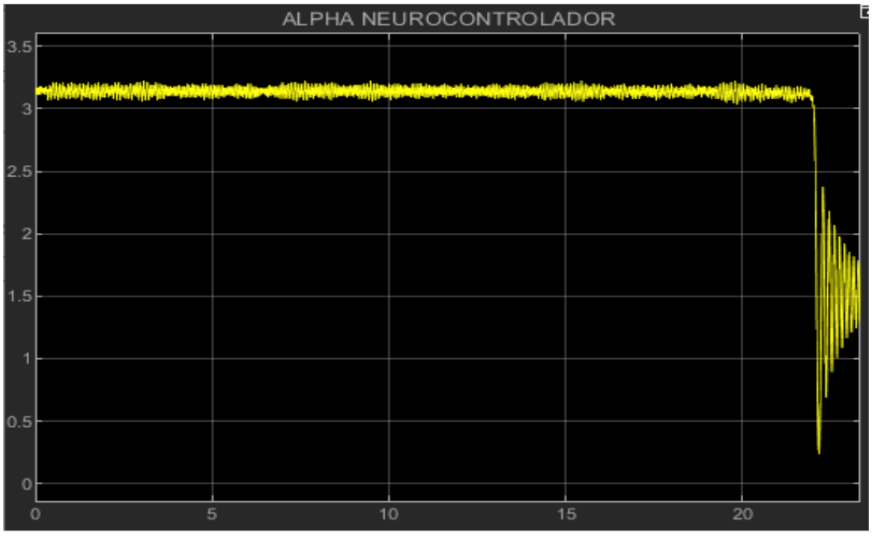

FIGURA 40: Alpha del neurocontrolador con tres retardos (Fuente: Autores).

9.2.4. Entrenamiento con 10 neuronas y 4 retardos eliminando la entrada de theta

Después de tener el control neuronal funcional se intentó reducir el número de entradas a la red. Se eliminó el ángulo theta de entrada, posteriormente se realizó el entrenamiento con el mismo set de datos con el que se obtuvo el controlador funcional. El MSE tuvo una diferencia notable siendo mayor el error, como se aprecia en la figura 41. A pesar de que se obtuvo un buen MSE, muy cercano a 0, se implementó el controlador pero se como se muestra en la figura 42 no logro controlar el ángulo del péndulo, por tal motivo esta entrada si es necesaria para la correcta predicción.

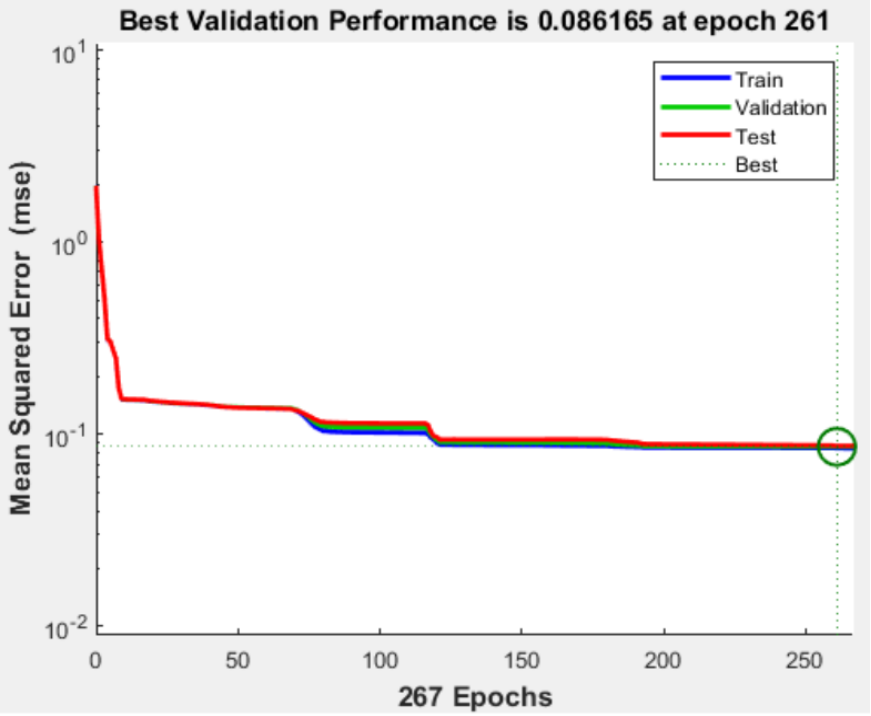

FIGURA 41: MSE sin theta inducido en la entrada (fuente: Autores).

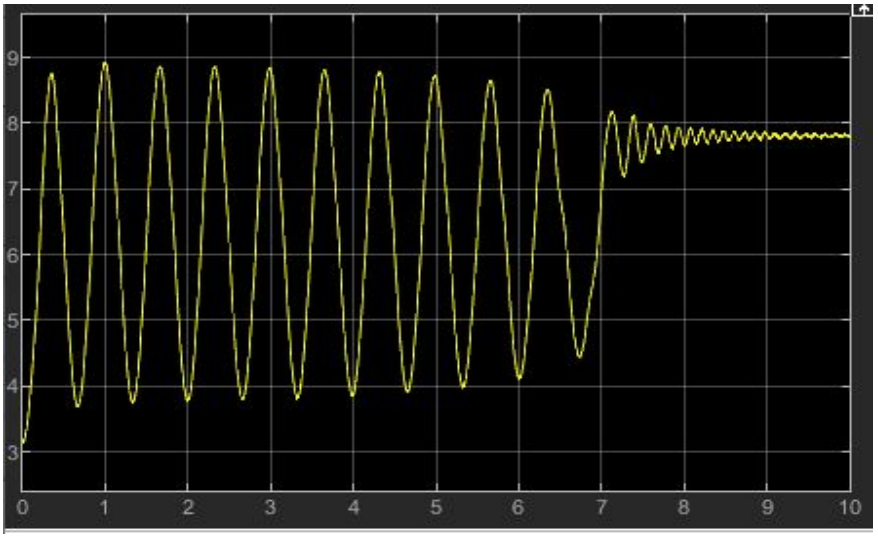

FIGURA 42: Alpha de neurocontrolador eliminando la entrada Theta (fuente: Autores).

9.2.5. Entrenamiento con 10 neuronas y 4 retardos eliminando la entrada del voltaje inducido

Se realizó el procedimiento descrito anteriormente pero eliminando la entrada de el voltaje inducido a la planta. En el entrenamiento se obtuvo un MSE muy bueno como se muestra en la figura 43, incluso mejor que en la arquitectura que se usaron 10 capas ocultas y 4 neuronas con todas las entradas de alpha, theta, el error y Vm, la cual fue la arquitectura con la que se obtuvieron los mejores resultados.

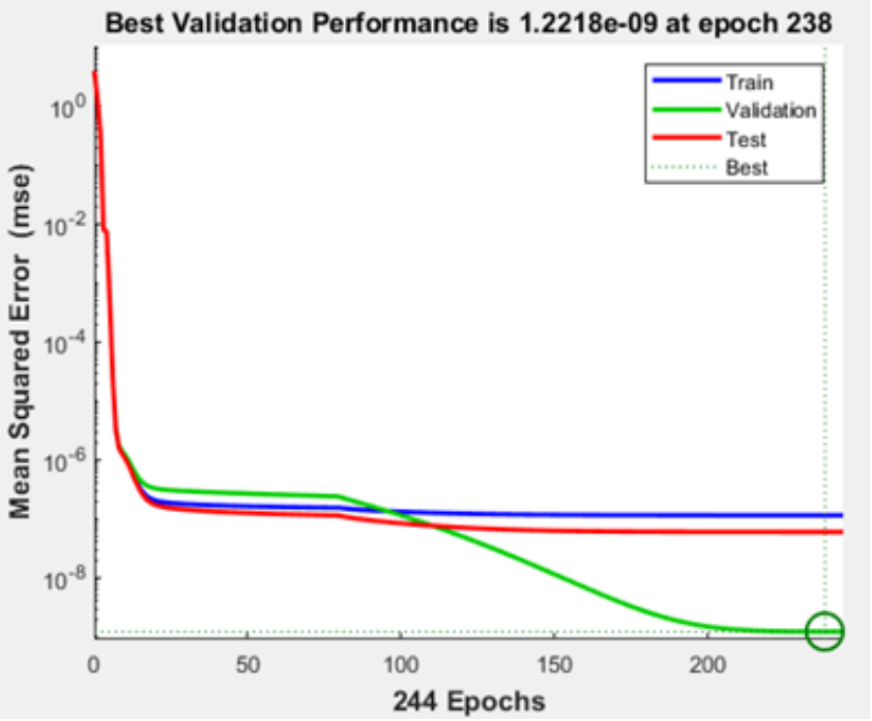

FIGURA 43: MSE del entrenamiento con 4 retardos y 10 neuronas en la capa oculta sin la entrada Vm (fuente: Autores).

Como se muestra en la figura 44, el tiempo que la red neuronal logro controlar el ángulo alpha del péndulo fueron aproximadamente 34 segundos, luego se desestabiliza.

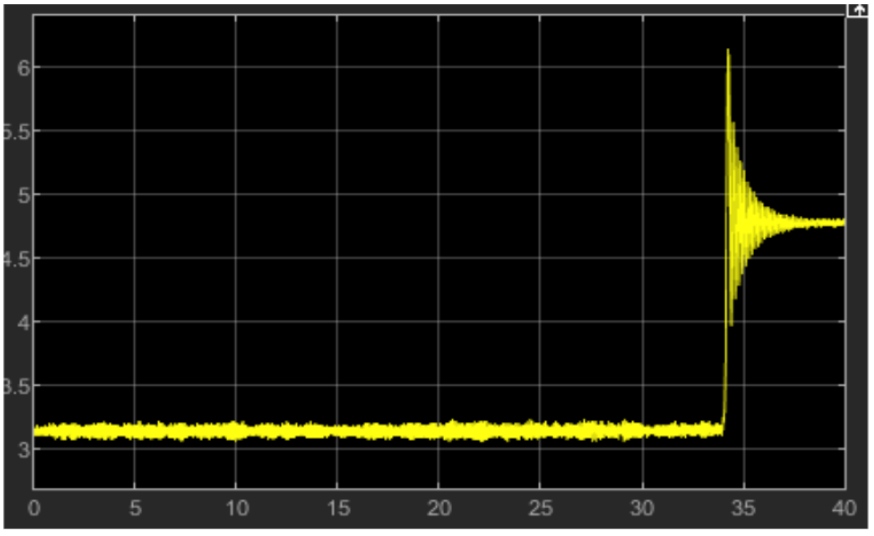

FIGURA 44: Ángulo alpha de la red sin la entrada Vm (fuente: Autores).

El cuadro 8 muestra los resultados obtenidos realizando las pruebas del control con las diferentes arquitecturas.

Arquitectura	**Error**	**Tiempo controlado**
Con 4 retardos y 10 Neuronas	$7{,}2 * 10^{-9}$	60 Segundos
Con 2 retardos y 10 Neuronas	$5{,}39 * 10^{-8}$	2 Segundos
Con 3 retardos y 10 Neuronas	$3{,}23 * 10^{-8}$	20 Segundos
Con 4 retardos, sin entrada Theta	0.086165	0 Segundos
Con 3 retardos, sin entrada de Voltaje	$1{,}2218 * 10^{-9}$	34 Segundos

CUADRO 8: Comparación entre las diferentes arquitecturas entrenadas (fuente: Autores).

De acuerdo con el cuadro que describe los resultados obtenidos con las diferentes arquitecturas se concluye que los mejores resultados se obtuvieron con 10 neuronas y 4 retardos como se muestra en la figura 30, si se usan menos de 4 retardos no es suficiente información para una buena predicción, pero aún así mantiene el péndulo en su punto de trabajo por poco tiempo; con 2 retardos logra estabilización solo 2 segundos, como se aprecia en la figura 36; con 3 retardos se estabilizó por 20 segundos (Figura 39). No se puede simplificar más entradas al controlador, puesto que si se llegara a quitar alguna entrada el MSE aumentaría, ya que no se proporciona la información necesaria para la identificación del péndulo invertido, aunque cuando se quitó la entrada del voltaje inducido también se tuvo un buen resultado y un MSE muy cercano a cero, el ángulo de theta empezaba a aumentar la velocidad con el tiempo por la razón de que no tenía manera cómo identificar el voltaje que le había inducido a la planta en momentos pasados, lo cual era información necesaria para el correcto control del péndulo.

Capítulo 10

Comparación del Controlador LQG con el Neurocontrolador Implementado

Se realizó una comparación del neurocontrolador que dio la mejor respuesta frente al control LQG aprovechando que en la librería del péndulo invertido se presenta el modelo en bloques de simulink (figura 45). La comparación se realiza evaluando tres parámetros: sobre impulso, error en estado estacionario y tiempo en el estado de estabilidad. Se realizó una simulación de un minuto de los controladores a estudiar estableciendo $180°$ en la condición inicial del ángulo de alpha de la planta.

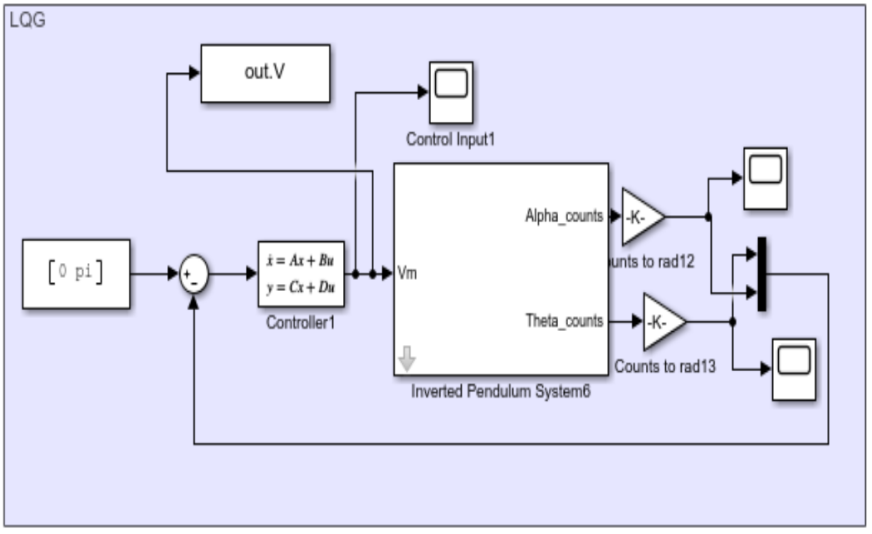

FIGURA 45: Controlador LQG para el péndulo invertido implementado en matlab [28].

10.1. Sobrepaso y error

La señal de alpha resultante de los controladores se muestra en la figura 46, la señal amarilla representa el ángulo del neurocontrolador y la azul el del controlador LQG.

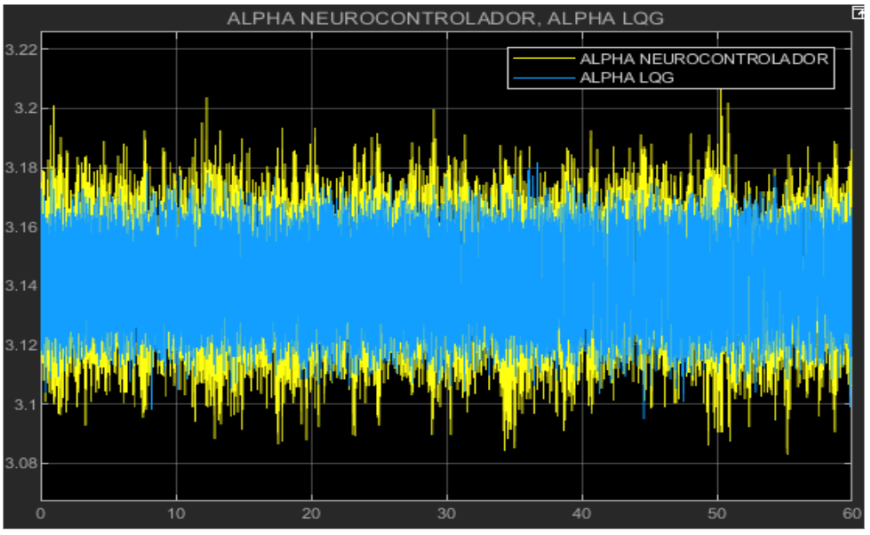

FIGURA 46: Comparación de alpha Neurocontrolador vs LQG (fuente: Autores).

En la gráfica 47 amarilla se presenta el error absoluto del ángulo de alfa del neurocontrolador y la azul es la del controlador LQG donde se ve que este último se aproxima más a 0.

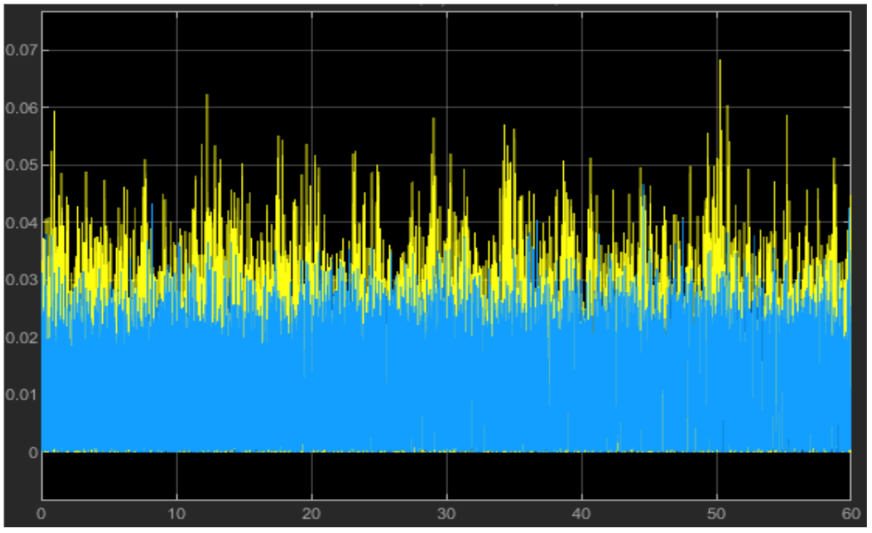

FIGURA 47: Comparación de error de alpha Neurocontrolador vs LQG (fuente: Autores).

Como la señal del error varía tanto en el tiempo se decidió promediar este valor haciendo uso de la función mean() de matlab, donde el controlador LQG presentó un error promedio de 0.0088 y un sobrepaso máximo de 0.043 en el segundo 8.223. Por otra parte el neurocontrolador tuvo un error promedio de 0.0239, una diferencia de 0.0158 respecto al LQG y un sobrepaso máximo de 0.07 en el segundo 51.

Los dos controladores logran equilibrar el péndulo el tiempo asignado, aunque como se muestra en la figura 48, el LQG lo hace de una mejor manera ya que el movimiento del ángulo de theta es mínimo, mientras que en el neurocontrolador este ángulo se comporta de una manera sinusoidal, moviéndose de izquierda a derecha. Este comportamiento era de esperarse pues en la identificación del sistema el voltaje eran dos señales sinusoidales.

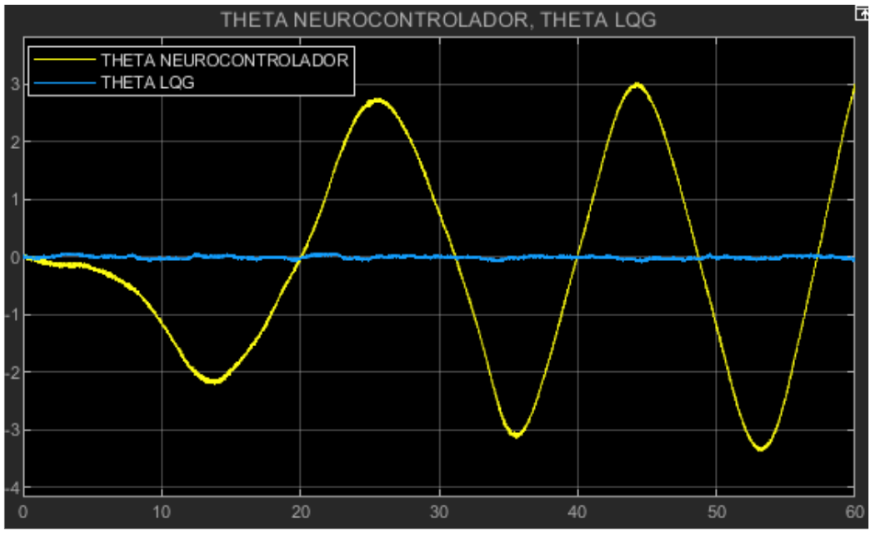

FIGURA 48: Comparación de theta Neurocontrolador vs LQG (fuente: Autores).

El control LQG no varía mucho en algulo thetha (Figura 48) mientras que el control neuronal permanece en constante movimiento, esto se debe a que en la identificación del sistema esta variable estaba en constante movimiento para lograr obtener un buen conjunto de datos como se explicó anteriormente, además, el control se diseñó solo para la alpha.

10.2. Respuesta a las perturbaciones

Se ingresaron varias perturbaciones que afectan el valor de la salida de alpha del péndulo, como se muestra en la figura 49 que se basa en modificar el valor de dicho ángulo mediante dos pasos en donde se puede modificar los parámetros de los bloques "step"para cambiar las propiedades

de la perturbación; su duración y la amplitud. En las pruebas que se realizaron se dejo una duración fija de 0.02s y se fue aumentando el valor de su amplitud. La simulación demora 10 s y en las figuras las gráficas amarillas representan las señales de salida del controlador LQG y las azules el neurocontrolador.

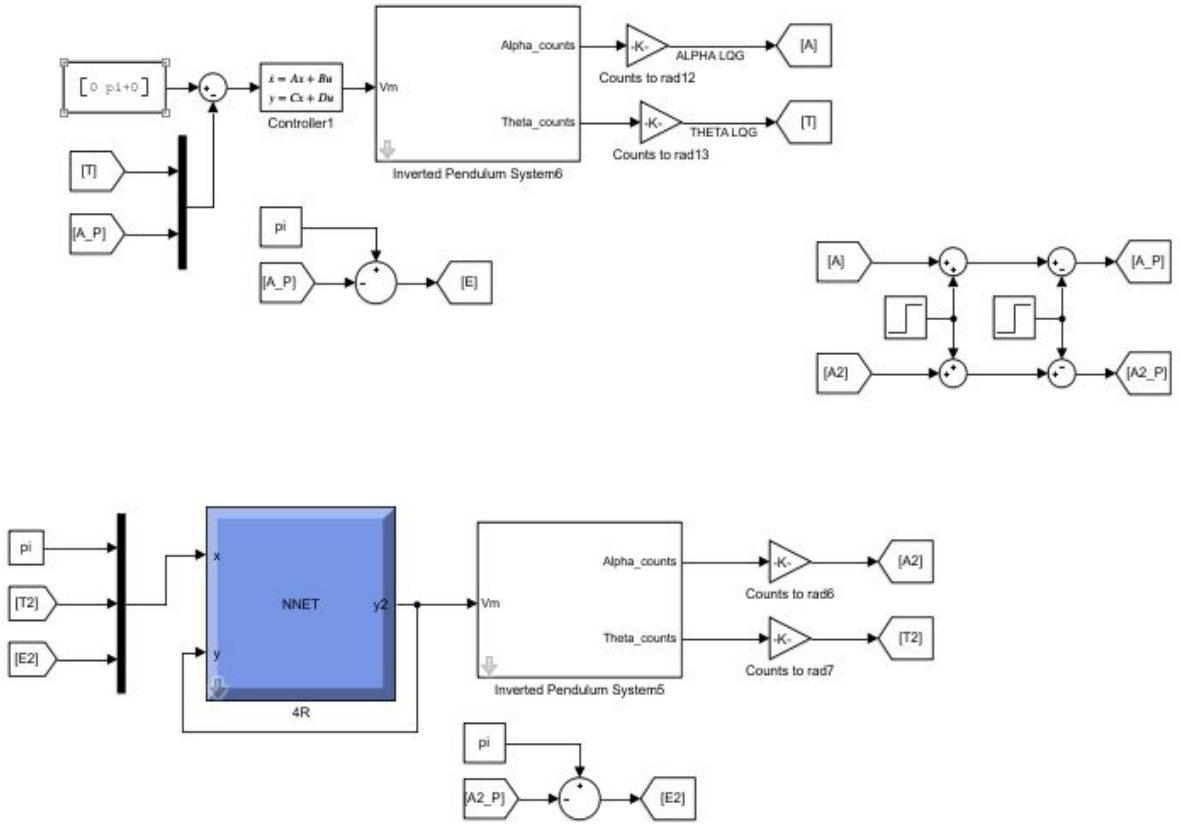

FIGURA 49: Simulación del neurocontrolador y el controlador LQG con perturbación (fuente: Autores).

10.2.1. Perturbación de amplitud de 0.4rad

Se ingresó una perturbación de 0.4rad en el segundo uno con una duración de 0.02s, en la figura 50 se muestra la señal de salida de alpha.

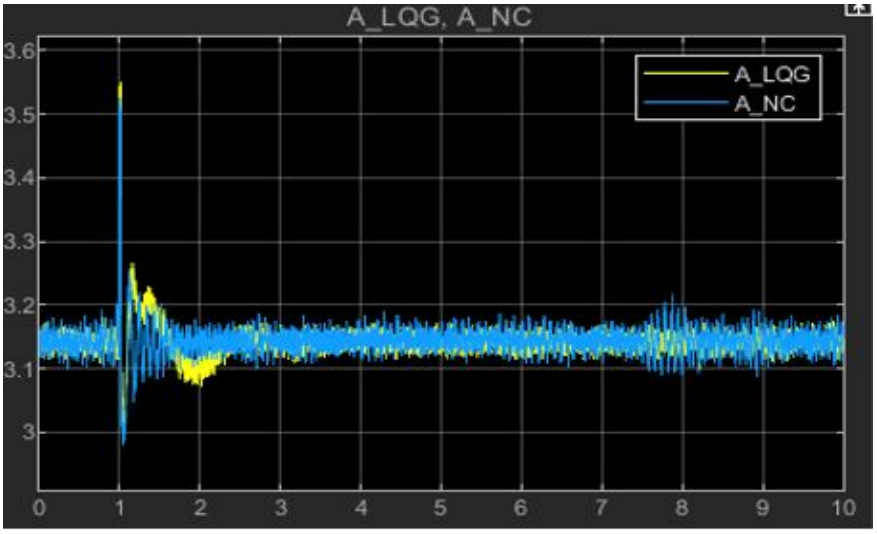

FIGURA 50: Alpha con perturbación de amplitud de 0.4rad y duración de 0.02s (fuente: Autores).

En la figura 51 se muestra el comportamiento de la salida de theta, se observa que en el neurocontrolador varia mucho esta salida luego de la perturbación pasando de 0rad a 3.8rad mientras que en el controlador LQG pasa de 0rad a -1.2rad.

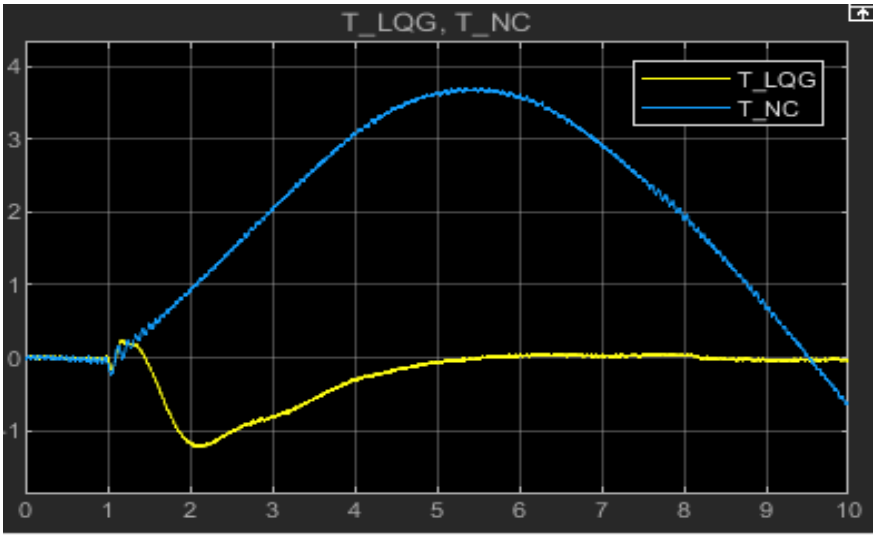

FIGURA 51: Theta con perturbación de amplitud de 0.4rad y duración de 0.02s (fuente: Autores).

En la figura 52 se muestra el error respecto al ángulo alpha, en este caso el tiempo de respuesta a la perturbación del neurocontrolador es menor que en el control LQG; el primero tiene un sobrepaso máximo de 0.15 y le toma aproximadamente 0.5s en reducir el valor del error promedio

al valor que toma cuando esta funcionando normalmente el controlador, es decir, sin perturbaciones y su estado inicial en el rango de trabajo, tal como se realizan las pruebas realizadas en la anterior sección. El segundo tarda aproximadamente 1.3s y tiene un sobrepaso máximo de 0.13.

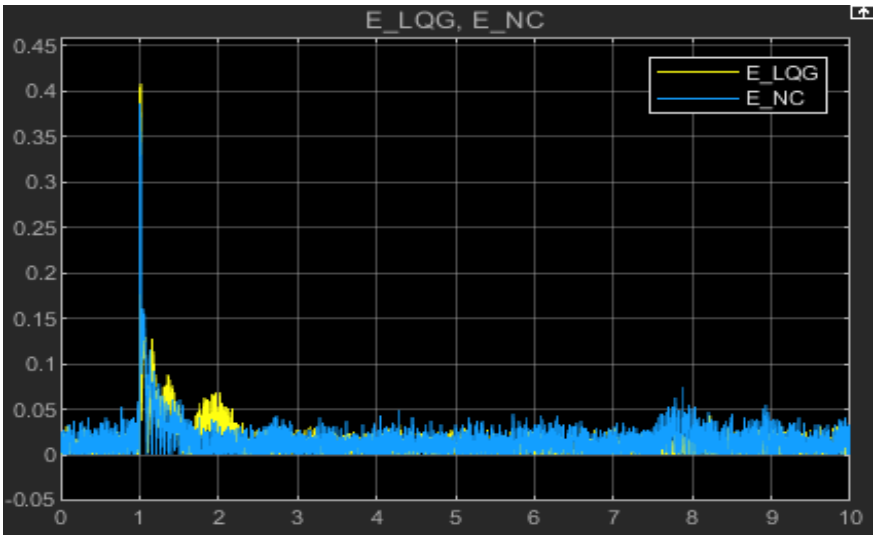

FIGURA 52: Error con perturbación de amplitud de 0.4rad y duración de 0.02s (fuente: Autores).

10.2.2. Perturbación de amplitud de 0.8rad

Se ingresó una perturbación de 0.8rad en el segundo uno con una duración de 0.02s, en la figura 53 se muestra la señal de salida de alpha.

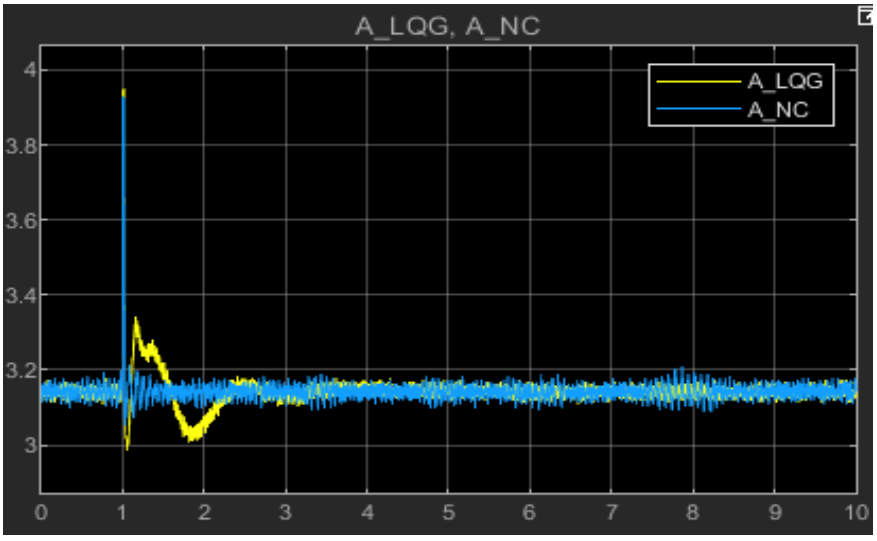

FIGURA 53: Alpha con perturbación de amplitud de 0.8rad y duración de 0.02s (fuente: Autores).

En la figura 54 se muestra el comportamiento de la salida de theta, se observa que en el neurocontrolador el error de esta salida llega hasta 2.3 en 8.8 s, mientras que en el controlador LQG llega al error que máximo (3) en 2s y vuelve a tener el control de la variable pasados los 4.4s después de la perturbación. El neurocontrolador no tiene control de la variable theta ya que solo esta diseñado para mantener la referencia de alpha.

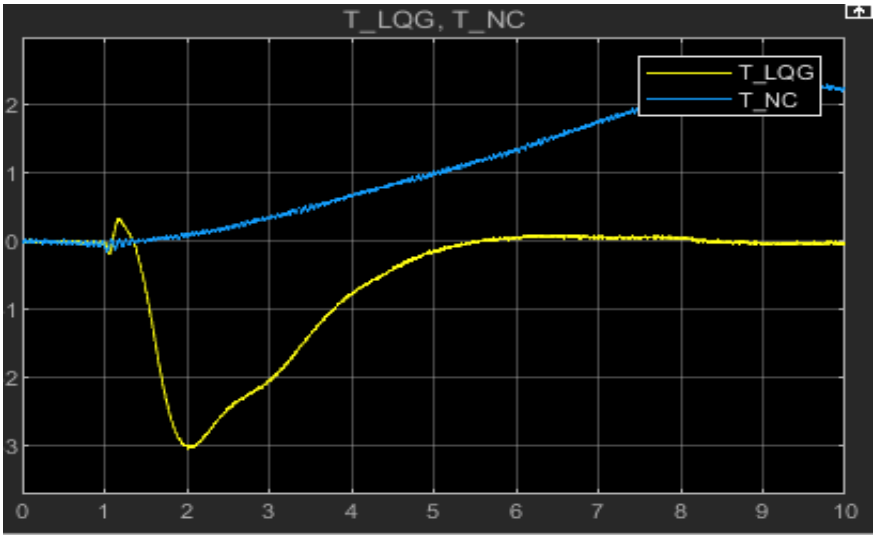

FIGURA 54: Theta con perturbación de amplitud de 0.8rad y duración de 0.02s (fuente: Autores).

En la figura 55 se muestra el error respecto al ángulo alpha, al igual que en el anterior caso, el tiempo de respuesta a la perturbación del neurocontrolador es menor que en el control LQG; el primero le toma aproximadamente 1.5s en reducir el valor del error promedio al valor que toma cuando esta funcionando normalmente el controlador. El segundo tarda aproximadamente 2.3s. El neurocontrolador tiene un sobrepaso máximo de 0.1 y el controlador LQG de 0.2

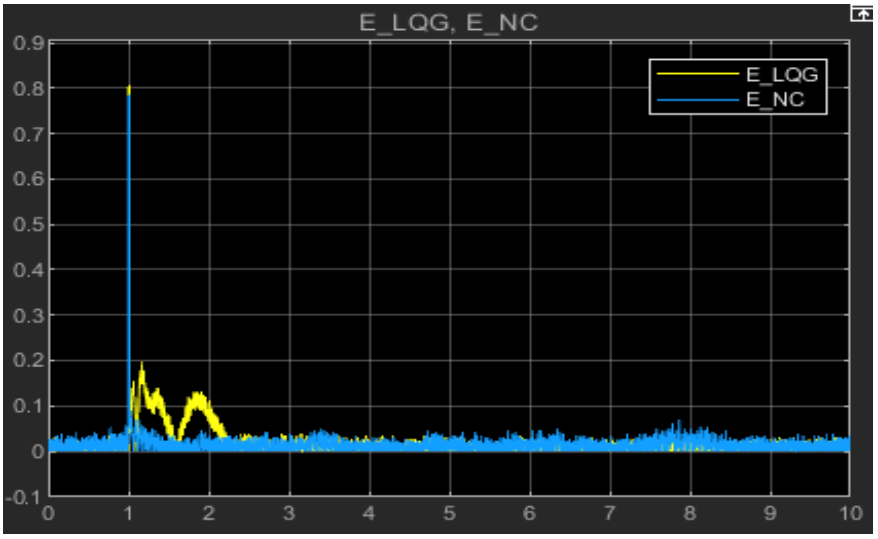

FIGURA 55: Error con perturbación de amplitud de 0.8rad y duración de 0.02s (fuente: Autores).

10.2.3. Perturbación de amplitud de 1.2rad

Se ingresó una perturbación de 1.2rad en el segundo uno con una duración de 0.02s, en la figura 56 se muestra la señal de salida de alpha.

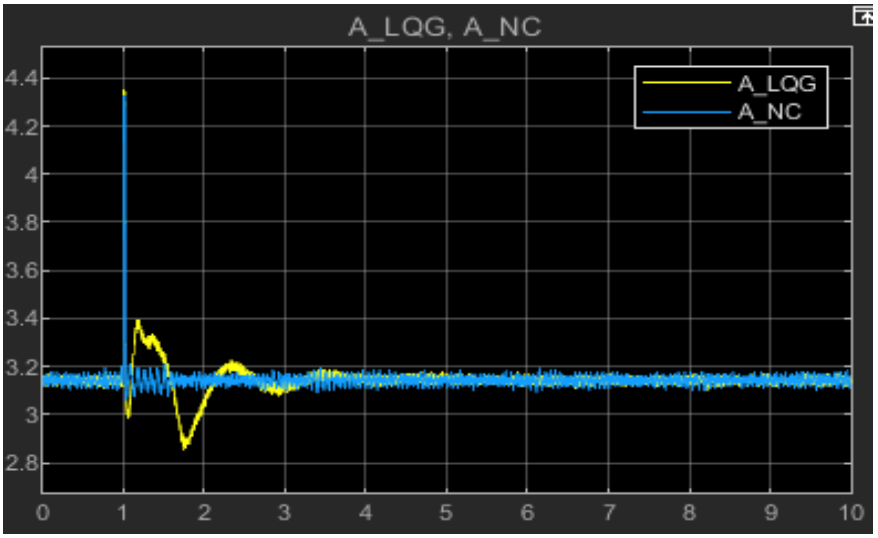

FIGURA 56: Alpha con perturbación de amplitud de 1.2rad y duración de 0.02s (fuente: Autores).

En la figura 57 se muestra el comportamiento de la salida de theta, se observa que en el neurocontrolador el error de esta salida llega como máximo a 2.3 en 8s, mientras que en el controlador LQG llega al error máximo (4.8) en 1.8s y vuelve a tener el control de la variable pasados los 4.2s después de la perturbación. El neurocontrolador no tiene control de la variable theta ya que solo esta diseñado para mantener la referencia de alpha.

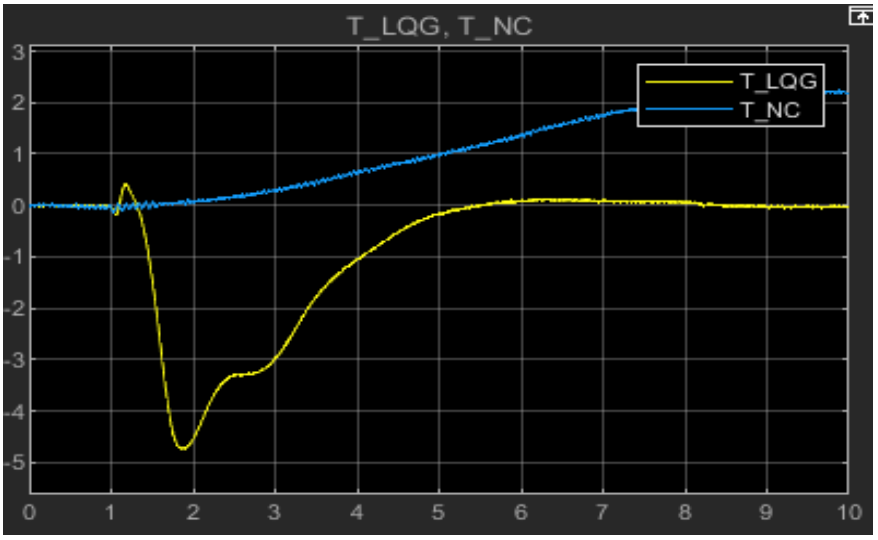

FIGURA 57: Theta con perturbación de amplitud de 1.2rad y duración de 0.02s (fuente: Autores).

En la figura 58 se muestra el error respecto al ángulo alpha, al igual que en el anterior caso, el tiempo de respuesta a la perturbación del neurocontrolador es menor que en el control LQG; el primero le toma aproximadamente 1.5s en reducir el valor del error promedio al valor que toma cuando esta funcionando normalmente el controlador. El segundo tarda aproximadamente 3s. El neurocontrolador tiene un sobrepaso máximo de 0.1 y el controlador LQG de 0.28.

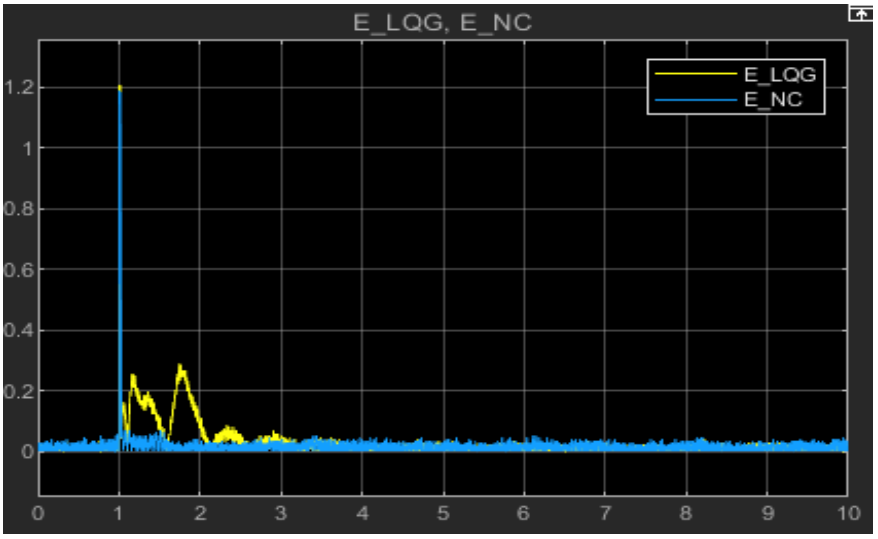

FIGURA 58: Error con perturbación de amplitud de 1.2rad y duración de 0.02s (fuente: Autores).

10.2.4. Perturbación de amplitud de 1.6rad

Se ingresó una perturbación de 1.6rad en el segundo uno con una duración de 0.02s, en la figura 59 se muestra la señal de salida de alpha. Se observa que el controlador LQG perdió el control con esta perturbación.

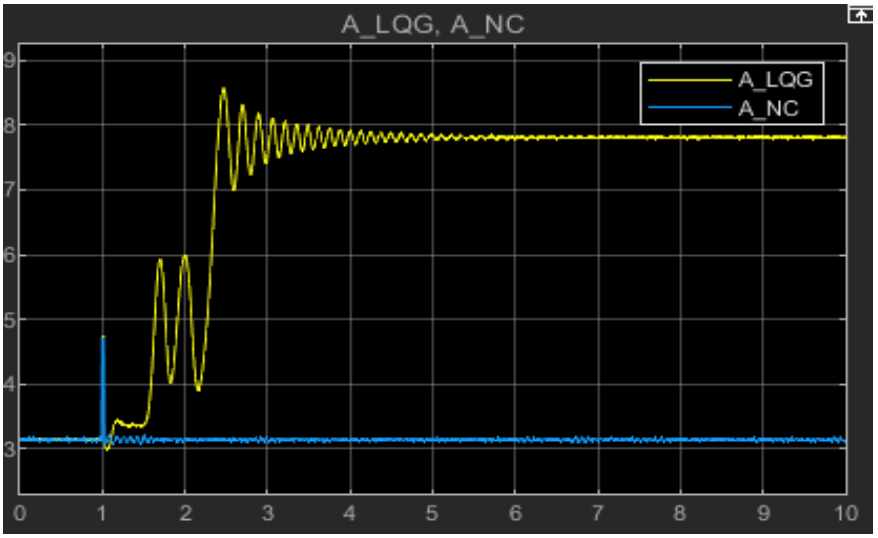

FIGURA 59: Alpha con perturbación de amplitud de 1.6rad y duración de 0.02s (fuente: Autores).

Cuando el controlador LQG pierde el control al igual que el neurocontrolador, no puede volver volver a levantar el péndulo y este empieza a girar en theta en una sola dirección como se muestra en la figura 60

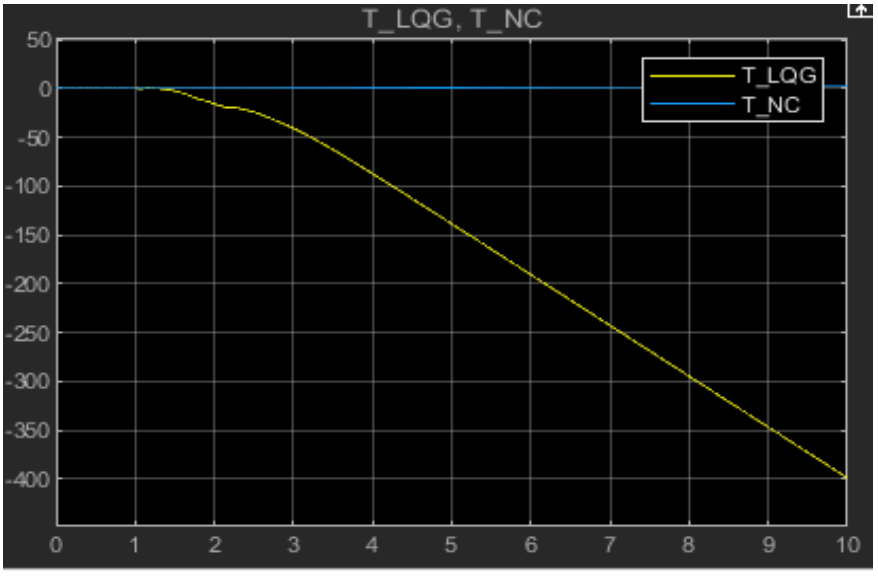

FIGURA 60: Theta con perturbación de amplitud de 1.6rad y duración de 0.02s (fuente: Autores).

En la figura 61 se visualiza la gráfica del error respecto a alpha, al igual que en los anteriores casos, a el neurocontrolador le toma aproximadamente 1.5s en reducir el valor del error promedio

al valor que toma cuando esta funcionando normalmente el controlador y tiene un sobrepaso máximo aproximadamente de 0.1.

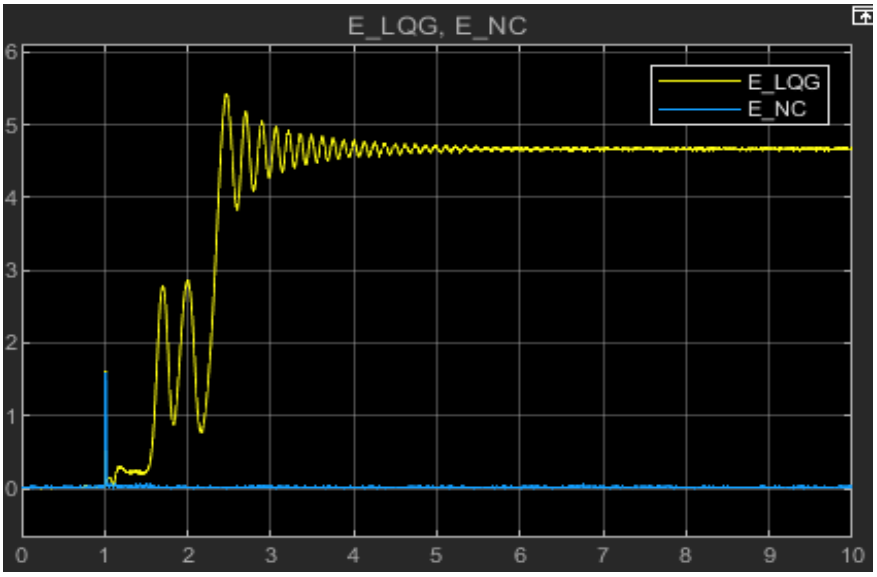

FIGURA 61: Error con perturbación de amplitud de 1.6rad y duración de 0.02s (fuente: Autores).

Se presenta en el cuadro 9 los datos de todas las perturbaciones, mostrando el sobrepaso máximo y el tiempo que tomo cada controlador en que el valor del error promedio sea menor o igual a cuando esta en estado estable . Los parámetros que se modificaron en las perturbaciones fue el valor de sus amplitudes, dejando una duración fija de 0.02 segundos con un tiempo de simulación de 10 segundos.

Perturbación (rad)	**Sobrepaso máximo (rad)**		**Tiempo (seg)**	
	NC	**LQG**	**NC**	**LQG**
Con amplitud de 0.4	0.15	0.13	0.5	1.3
Con amplitud de 0.8	0.1	0.2	1.5	2.3
Con amplitud de 1.2	0.1	0.28	1.5	3
Con amplitud de 1.6	0.1	NO CONTROLA	1.5	NO CONTROLA

CUADRO 9: Comparación de controladores ante perturbaciones (fuente: Autores).

Capítulo 11

Discusión

De acuerdo con las pruebas y análisis realizados se infiere que la arquitectura apropiada para que la red logre la identificación de la inversa de la planta es con 4 retardos en la entrada, al hacer uso de mas o menos retardos baja el desempeño del controlador, disminuyendo el tiempo en que es capaz de estabilizar el péndulo o simplemente no controla. El numero de retardos es directamente proporcional al tiempo de entrenamiento. El número de neuronas en la capa oculta se selecciono de 10 se observo que al aumentar este numero de neuronas el tiempo de entrenamiento en cada época aumentaba demasiado pero se aproximaba mas ràpido a 0 el MSE, mientras que cuando se disminuye este parámetro el tiempo de entrenamiento por epoca es menor pero se demora mas en aproximarse a 0, al final de cuentas llegaba al mismo error pero lo que cambia es el tiempo en el que llega a ese valor, entonces lo que se busca es un numero de neuronas apropiado para que no se demore tanto el entrenamiento y el error al final sea lo mas aproximado a 0.

A partir de las pruebas realizadas en la comparación de el desempeño de los controladores se observo que cuando existen perturbaciones de alta amplitud la señal de theta del controlador LQG se ve afectada de una manera brusca en comparación del neurocontrolador debido a que este fue diseñado solo para controlar alpha por lo cual presentó una respuesta mas ràpida a la perturbación y un sobrepaso menor a esta. Además la manera como se obtuvieron los datos para el entrenamiento de la RNN favorece las buenas respuestas a las perturbaciones ya que se buscaban valores aleatorios en la señal de entrada para que el péndulo estuviera el mayor tiempo posible en el rango de trabajo y así obtener un conjunto de datos que lograra identificar la inversa de la planta. El control LQG no varía mucho el algulo thetha mientras que el control neuronal permanece en constante movimiento, esto se debe a que en la identificación del sistema esta variable estaba en constante movimiento para lograr obtener un buen conjunto

de datos como se explicó anteriormente, además, el control se diseño solo para alpha, mientras que el control LQG si busca que theta permanezca en 0,

Capítulo 12

Conclusiones

Para la selección del número de capas y neuronas no existe un método analítico que permita definir dichos parámetros, no existen métodos establecidos para el diseño de una RN por tanto, en este trabajo el criterio utilizado fue la experiencia adquirida y las investigaciones de otros autores, permitiendo por medio de la práctica de ensayo y error la adecuación de la red que mejor se ajustara a la solución del problema. La RN que nos dió la mejor solución está constituida por 3 capas; la de entradas, una oculta con 10 neuronas y finalmente la de salida con una función lineal.

En la obtención de los patrones de entrenamiento el tiempo de muestreo es un parámetro crítico, se necesitan suficientes ejemplos para entrenar una RN, de lo contrario, habrá valores que la RN no podrá reconocer y dar salidas inesperadas. Por el contrario, si existen demasiados datos la RN puede necesitar muchas iteraciones para poder entrenarse y buscar optimizar los pesos. La asignación de los retardos en la capa de entradas también es un parámetro fundamental ya que este depende de que tanta información pasada necesita para lograr una correcta predicción.

Con el algoritmo Levenberg Marquardt la búsqueda por los mínimos es local por lo que puede fracasar en acercarse al mínimo global y nunca aproximarse en su dirección correctamente. Cuando el entrenamiento cae en un mínimo local sin satisfacer el porcentaje de error se puede considerar: cambiar la topología de la red (número de capas y número de neuronas), comenzar el entrenamiento con pesos iniciales diferentes, modificar los parámetros de aprendizaje, modificar el conjunto de entrenamiento o presentar los patrones en otro orden.

Se puede apreciar que la relación entre el número de capas, neuronas, pesos y demás parámetros de entrenamiento pueden llegar a ser independiente entre sí, no siempre la red neuronal

con el mayor número de neuronas en la capa oculta será la que entregue los mejores resultados, como tampoco lo será el que más o menos retardos tenga, como se muestra en el cuadro 8, por eso es importante tener de base un conjunto de datos que brinde la información necesaria y tratar de llevar al ensayo diversas configuraciones de redes neuronales para apreciar su comportamiento.

A partir de las gráficas comparativas obtenidas entre el control LQG y la red neuronal, se evidencia que el desempeño del control clásico presenta un mejor comportamiento respecto al neurocontrolador diseñado cuando no hay perturbaciones [46], [47], [48], pero cuando las hay la red neuronal presenta una mejor respuesta a estas, tal como se presenta en el cuadro 9.

Cuando existen perturbaciones de alta amplitud la señal de theta del controlador LQG se ve afectada de una manera brusca en comparación del neurocontrolador debido a que este fue diseñado solo para controlar alpha por lo cual presentó una respuesta mas rápida a la perturbación y un sobrepaso menor a esta, como se muestra en el cuadro 9. Además la manera como se obtuvieron los datos para el entrenamiento de la RNN favorece las buenas respuestas a las perturbaciones ya que se buscaban valores aleatorios en la señal de entrada para que el péndulo estuviera el mayor tiempo posible en el rango de trabajo.

Aunque la comparación muestra que la red neuronal presenta un desempeño más bajo frente al método de control clásico cuando se realizaron las pruebas sin perturbaciones, no significa que las técnicas de aprendizaje de máquina no sirvan o no puedan llegar a presentar mejor desempeño, ya que en los artículos revisado en el estado del arte algunas redes implementadas superan en desempeño frente a controles clásicos.

Capítulo 13

Trabajos Futuros

A continuación, se presentan algunos trabajos futuros que podrían surgir como resultado del presente proyecto:

En primer lugar, se podría implementar entrenamientos con otras tipologías de redes neuronales, como redes LST que en muchos casos puede llegar a dar resultados mejores. También se podría intentar mejorar el sistema actual con otras configuraciones, cambiando el número de capas ocultas, cambiando el set de datos o modificando otros parámetros.

A partir de esta misma metodología se podría escalar el proyecto a la implementación de un sistema de péndulos dobles. Mejorando el entrenamiento del sistema ya implementado.

Bibliografía

[1] Ethem Alpaydin. *Introduction to Machine Learning*. 4.ª ed. MIT Press Academic, 2020.

[2] María M. Seron. «Sistemas No Lineales». En: *Universidad Nacional de Rosario* (2000), págs. 7-8.

[3] M. J. G. Guarnizo, R. C. L. Trujillo y M. J. A. Guacaneme. «Modeling and control of a two DOF helicopter using a robust control design based on DK iteration». En: *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*. 2010, págs. 162-167.

[4] Gerson. Figueroa. «Análisis de un péndulo invertido mediante técnicas de control inteligente». En: *Instituo Politécnico Nacional* (2013), págs. 1-172.

[5] Geun Hyeong y Seul Jung. «Control of inverted pendulum system using a neuro-fuzzy controller for intelligent control education». En: *2008 IEEE International Conference on Mechatronics and Automation*. IEEE, 2008, págs. 965-970.

[6] Alexander Winkler y Jozef Suchý. «Identification and controller design for the inverted pendulum actuated by a position controlled robot». En: *18th International Conference on Methods Models in Automation Robotics (MMAR)"*. IEEE, 2013, págs. 285-263.

[7] Richard F Chidzonga y Fredson. A. Phiri. «Stabilizing an inverted pendulum on an airbed track». En: *AFRICON Conference 2007*. IEEE, 2007, págs. 1-7.

[8] Javier Aracil y Francisco Gordillo. «The inverted pendulum: a benchmark in nonlinear control». En: *Proceedings World Automation Congress, 2004.* IEEE, 2004, págs. 468-482.

[9] Nenad Muskinja y Boris Tovornik. «Swinging up and stabilization of a real inverted pendulum». En: *IEEE Transactions on Industrial Electronics* 53.2 (2006), págs. 631-639.

[10] Warren N. White y col. «Design, build, and test of an autonomous inverted pendulum cart». En: *2013 American Control Conference*. IEEE, 2013, págs. 5893-5898.

[11] Khizir Mahmud. «Design and analysis of the control of an inverted pendulum system by Matlab». En: *2013 IEEE Global High Tech Congress on Electronics*. IEEE, 2013, págs. 207-211.

[12] Samatthachai Panya y col. «Hybrid Controller for Inverted Pendulum System». En: *2008 International Symposium on Communications and Information Technologies (ISCIT)*. IEEE, 2008, págs. 385-388.

[13] Ju-Bong Kim y col. «Imitation Reinforcement Learning-Based Remote Rotary Inverted Pendulum Control in OpenFlow Network». En: *IEEE Access* 7 (2019), págs. 36682-36690.

[14] Takanori Shibata y col. «Skill based control by using fuzzy neural network for hierarchical intelligent control». En: *1992 IJCNN International Joint Conference on Neural Networks*. IEEE, 1992, págs. 81-86.

[15] Bavarian B. «Introduction to neural networks for intelligent control». En: *IEEE Control Systems Magazine* 8.2 (1988), págs. 3-7.

[16] Kumpati S Narendra y Snehasis Mukhopadhyay. «Intelligent control using neural networks». En: *IEEE Control Systems Magazine* 12.2 (1992), págs. 11-18.

[17] Chrysostomos Stylios y Peter Groumpos. «Fuzzy cognitive maps: a soft computing technique for intelligent control». En: *2000 IEEE International Symposium on Intelligent Control. Held jointly with the 8th IEEE Mediterranean Conference on Control and Automation (Cat. No.00CH37147)*. IEEE, 2000, págs. 97-102.

[18] Jyoti Krishen y Victor M Becerra. «Efficient fuzzy control of a rotary inverted pendulum based on LQR mapping». En: *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*. IEEE, 2006, págs. 2701-2706.

[19] Ammar Al-Jodah, Hassan Zargarzadeh y Maythem K Abbas. «Experimental verification and comparison of different stabilizing controllers for a rotary inverted pendulum». En: *2013 IEEE International Conference on Control System, Computing and Engineering*. IEEE, 2013, págs. 417-423.

[20] Monica Roman, Eugen Bobasu y Dorin Sendrescu. «Modelling of the rotary inverted pendulum system». En: *2008 IEEE International Conference on Automation, Quality and Testing, Robotics*. IEEE, 2008, págs. 141-146.

[21] P. Karam J. Apkarian y M. Lévis. «Rotary Inverted Pendulum Experiment for MATLAB ® /Simulink ® Users». En: *Quanser Inc.* (2012), págs. 1-6.

[22] Tianxiao Liu. «U.S. Pandemic Prediction Using Regression and Neural Network Models». En: *2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*. IEEE, 2020, págs. 351-354.

[23] Tushar Verma y col. «Data Analysis to Generate Models Based on Neural Network and Regression for Solar Power Generation Forecasting». En: *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*. IEEE, 2016, págs. 97-100.

[24] Aakanksha Sharaff y Samuel Robin Roy. «Comparative Analysis of Temperature Prediction Using Regression Methods and Back Propagation Neural Network». En: *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 2018, págs. 739-742.

[25] R. C. L. Trujillo M. J. G. Guarnizo y M. J. A. Guacaneme. «General Inverse Neural Current Control for Buck Converter». En: *Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics. Springer, Dordrech* (2008), págs. 117-122.

[26] Razeef Mohd, Muheet Ahmed Butt y Majid Zaman Baba. «SALM-NARX: Self Adaptive LM- based NARX model for the prediction of rainfall». En: *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on*. IEEE, 2018, págs. 580-585.

[27] Osorio Zúñiga Carlos Andrés. «DISEÑO, CONSTRUCCION Y CONTROL DE UN PENDULO INVERTIDO ROTACIONAL UTILIZANDO TECNICAS LINEALES Y NO LINEALES». En: *Universidad Nacional de Colombia* (2009), págs. 13-14.

[28] Hill Rick. «Inverted Pendulum Control». En: *MATLAB Central File Exchange* (2013).

[29] Atria Innovation. *Qué son las redes neuronales y sus funciones*. Oct. de 2019. URL: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/#:~:text=Las-redes-neuronales-artificiales-son,entrada20hasta-generar-una-salida.

[30] C. Luis y M. Diego. «Redes Neuronales». En: *Cinvestav* (2020).

[31] Interactive Chaos. *Backpropagation | Interactive Chaos*. URL: https://interactivechaos.com/es/manual/tutorial-de-machine-learning/backpropagation.

[32] Isis Bonet y col. «Redes neuronales recurrentes para el análisis de secuencias». En: *2007 Revista Cubana de Ciencias Informáticas* 1.2 (2007), págs. 48-57.

[33] Jordi Torres. *Redes Neuronales Recurrentes*. Mar. de 2021. URL: https://torres.ai/redes-neuronales-recurrentes/.

[34] Galán Pablo. «INCORPORACIÓN DE INTELIGENCIA ARTIFICIAL EN GEMELO VIRTUAL BÁSICO DE ROBOT INDUSTRIAL». En: *UNIVERSIDAD DE CANTABRIA* (2019).

[35] Calvo Diego. «Red Neuronal Recurrente – RNN». En: *Diego Calvo* (2018).

[36] R. Aguilar y A. Martín. «RED NEURONAL AUTORREGRESIVA NO LINEAL CON ENTRADAS EXÓGENAS PARA LA PREDICCIÓN DEL ELECTROENCEFALOGRAMA FETAL». En: *Department of Computer and Systems Engineering, University of La Laguna* (2017), pág. 2.

[37] Victor Rodriguez, Jaime Garzón y Jesús López. «Control Neuronal por Modelo Inverso de un Servosistema Usando Algoritmos de Aprendizaje Levenberg-Marquardt y Bayesiano». En: *VIII Congreso de la Asociación Colombiana de Automática.* Universidad Tecnológica de Bolívar, 2009, págs. 1-6.

[38] Jaime Durán. *Todo lo que Necesitas Saber sobre el Descenso del Gradiente Aplicado a Redes Neuronales*. Sep. de 2019. URL: https://medium.com/metadatos/todo-lo-que-necesitas-saber-sobre-el-descenso-del-gradiente-aplicado-a-redes-neuronales-19bdbb706a78.

[39] Sergio Ledesma. «Las Redes Neuronales implementación y consideraciones prácticas». En: *Fifth Mexican International Conference on Artificial Intelligence*. Universidad de Guanajuato, 2016, págs. 1-50.

[40] Jose Martinez Heras. *Error Cuadrático Medio para Regresión*. Oct. de 2020. URL: https://www.iartificial.net/error-cuadratico-medio-para-regresion/#:~:text=El-Error-Cuadr%C3%A1tico-Medio-es,podremos-indicar-el-resultado-correcto..

[41] IBM. *Acerca de la regresión lineal*. 2020. URL: https://www.ibm.com/co-es/analytics/learn/linear-regression#:~:text=%C2%BFQu%C3%A9-es-la-regresi%C3%B3n-lineal,variable-se-denomina-variable-independiente..

[42] Luis Garrido. «Identificación, estimación y control de sistemas no-lineales mediante RGO». En: *Universidad Carlos III de Madrid* (1999).

[43] National Instruments. *Adquirir una Señal Analógica: Ancho de Banda, Teorema de Muestreo de Nyquist y Aliasing*. 2019. URL: https://www.ni.com/es-co/innovations/white-papers/06/acquiring-an-analog-signal--bandwidth--nyquist-sampling-theorem-.html.

[44] U. Francisco. «Estrategia de identificación dinámica no lineal basada en NARX para fuentes de generación distribuida acompladas electrónicamente a Micro-Redes AC/DC». En: *Universidad Politécnica Salesina - SEDE QUITO* (2020), pág. 34.

[45] M. Moumouni Hamidou y N. Talibi Soumaïla. «Electrical Charge of Niamey City Modelisation by Neural Network». En: *Sci. J. Energy Eng., vol. 7* (2019), pág. 13.

[46] Zina Boussaada y col. «A nonlinear autoregressive exogenous (NARX) neural network model for the prediction of the daily direct solar radiation». En: *The 10th International Conference on Sustainable Energy and Environmental Protection*. Vol. 11. 3. IEEE, 2018, pág. 620.

[47] Luis Gonzaga Baca Ruiz y col. «An Application of Non-Linear Autoregressive Neural Networks to Predict Energy Consumption in Public Buildings». En: *Energies* 9.9 (2016).

[48] Casimiro Rocha y Josér Escorcia. «Sistema de Visión Artificial para la Detección y el Reconocimiento de Señales de Tráfico Basado en Redes Neuronales». En: *2010 LACCEI Latin American and Caribbean Conference for Engineering and Technology*. 2010.

[49] Llano. L y Hoyos. A. «Comparación del Desempeño de Funciones de Activación en Redes Feedforward para aproximar Funciones de Datos con y sin Ruido.» En: *Universidad Nacional de Colombia* (2007).

[50] I. Maldonado y A. Zabidi. «Estimación de los parámetros de un modelo haciendo uso de correspondencias con incertidumbre». En: *Centro de Investigacion en Matematicas, A.C - CIMAT* (2011), págs. 20-25.

[51] Nadiah Mohamad y col. «Comparison between Levenberg-Marquardt and Scaled Conjugate Gradient training algorithms for Breast Cancer Diagnosis using MLP». En: *6th International Colloquium on Signal Processing its Applications*. IEEE, 2010, págs. 1-7.

[52] J. Maria y R. Diego. «Desarrollo y evaluación de un control Neuro-Difuso tipo ANFIS frente a un control PID convencional aplicado al péndulo invertido». En: *Universidad Politécnica Salesiana* (2016).

[53] k. Maher. «Estudio comparativo entre el control PID Clásico y el Control PID Fraccionario alpicado al sistema del péndulo invertido». En: *Universidad Técnológica de Pereria* (2020), págs. 98-106.

[54] Khushboo Barya, Sheela Tiwari y Rameshwar Jha. «Comparison of LQR and robust controllers for stabilizing inverted pendulum system». En: *2010 INTERNATIONAL CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING TECHNOLOGIES*. IEEE, 2010, págs. 300-304.

[55] Faiber Robayo, Ana Barrera y Laura Polanco. «Desarrollo de un controlador basado en redes neuronales para un sistema multivariable de nivel y caudal». En: *Revista Ingeniería y Región* 14.2 (2015), págs. 43-54.