PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

CONTROL DE UN SISTEMA DE POSICIONAMIENTO MAGNÉTICO DE DOS DIMENSIONES USANDO APRENDIZAJE PROFUNDO POR REFUERZO

Tesis pata optar el grado de Magíster en Ingeniería de Control y Automatización que presenta:

Eduardo Alberto Martín Bejar Espejo

ASESOR: Antonio Manuel Morán Cárdenas

Octubre, 2018

Resumen

Los sistemas de posicionamiento magnético son preferidos respecto a sus contrapartes mecánicas en aplicaciones que requieren posicionamiento de alta precisión como en el caso de la manufactura de circuitos integrados. Esto se debe a que los actuadores electromagnéticos no sufren los efectos de la fricción seca o desgaste mecánico. Sin embargo, estos sistemas poseen fuertes no linealidades que dificultan la tarea de control. Por otro lado, el aprendizaje por refuerzo se ha posicionado como una técnica de entrenamiento de redes neuronales prometedora que está permitiendo resolver varios problemas complejos. Por ejemplo, el aprendizaje por refuerzo fue capaz de entrenar redes neuronales que han logrado vencer al campeón mundial de Go, derrotar a varios jugadores profesionales de ajedrez y aprender a jugar varios videojuegos de la consola Atari. Asimismo, estas redes neuronales están permitiendo la manipulación de objetos por brazos robóticos, un problema que era muy difícil de resolver por medio de técnicas tradicionales. Por esta razón, el presente trabajo tiene como objetivo diseñar un controlador neuronal entrenado por refuerzo para el control de un sistema de posicionamiento magnético de dos dimensiones. Se utiliza una variación del algoritmo *Deep Deterministic Policy Gradient* (DDPG) para el entrenamiento del controlador neuronal. Los resultados obtenidos muestran que el controlador diseñado es capaz de alcanzar varios setpoints asignados y de realizar el seguimiento de una trayectoria dada.

Índice

Índice de figuras

Índice de tablas

Índice de algoritmos

Introducción

En el transcurso de los últimos años la inteligencia artificial ha experimentado una revolución nunca antes vista debida a los numerosos avances en el desarrollo de algoritmos basados en redes neuronales que aprovechan la elevada potencia de cómputo disponible hoy en día [10]. En particular, la compañía británica Deepmind, adquirida por Google en el año 2014, es uno de los principales protagonistas de esta revolución. En sus más recientes trabajos, Deepmind ha creado una inteligencia artificial general llamada Alpha Zero que se enseña así misma a jugar los juegos de mesa Ajedrez y Go sin ningún tipo de conocimiento previo programado y que es actualmente invencible [11], [12]. Las inteligencias artificiales creadas por Deepmind consisten en redes neuronales, que son algoritmos computacionales los cuales están inspirados en el funcionamiento del cerebro humano y que consisten de diversos parámetros que hay que ajustar adecuadamente para resolver una tarea de manera óptima. Al proceso de encontrar estos parámetros se le conoce como el aprendizaje de la red neuronal y es una de las principales problemáticas al usar redes neuronales. El aporte de Deepmind es el de usar un método de aprendizaje inspirado en la forma en la que los humanos aprendemos a solucionar problemas: por prueba y error. A este tipo de aprendizaje se le conoce dentro del campo de la psicología como aprendizaje por refuerzo [13].

Las redes neuronales son la técnica más efectiva con la que cuenta la inteligencia artificial hoy en día y que se ha caracterizado por presentar una evolución lenta [14]. Por ejemplo, desde que Donald Hebb propusiera un modelo matemático que pretendía explicar el mecanismo de plasticidad de las neuronas [15], tuvieron que pasar casi dos décadas para que recién en 1958 Frank Rosenblatt presentara el primer modelo de una red neuronal [16]. Asimismo, no fue hasta el año 1975 en que Paul Werbos introdujo el concepto de algoritmo de aprendizaje que permite a las redes neuronales auto

configurarse para resolver una tarea de manera óptima [17]. A partir de ese momento se empezó a usar redes neuronales para resolver problemas en reconocimiento de patrones y control. Sin embargo, fueron muy pocas las aplicaciones en las que las redes neuronales superaron en efectividad a las técnicas tradicionales de procesamiento de señales y control.

El panorama de hoy en día es completamente distinto ya que las redes neuronales son el estado del arte en prácticamente cualquier aplicación que requiera el procesamiento de una señal digital como texto, audio e imágenes [18]. Asimismo, las redes neuronales permiten actualemente resolver tareas que se consideraban extremadamente difíciles de resolver en robótica como la manipulación de objetos por medio de brazos robóticos [19].

El punto de inflexión en el campo de la inteligencia artificial como la conocemos hoy en día se dió en 2012 cuando investigadores de la Universidad de Toronto, liderados por Geoffrey Hinton, presentaron un algoritmo computacionalmente eficiente que permitía entrenar redes neuronales profundas [20], es decir, redes neuronales que cuentan con un número extremadamente alto de neuronas frente a las que se usaban hasta esa fecha. Tal algoritmo fue aplicado para clasificar imágenes en el conjunto de datos de ImageNet, la referencia mundial para evaluar la efectividad de algoritmos de clasificación de imágenes y que es usado por investigadores de todo el mundo. Los resultados del grupo de Geoffrey Hinton fueron sorprendentes ya que su método logró un puntaje de efectividad en clasificación de imágenes de cerca del 95%, el más alto hasta esa fecha y comparable con el puntaje alcanzado por seres humanos. Después de este hecho, la investigación en redes neuronales se incrementó drásticamente [14].

No paso mucho tiempo para que Deepmind publicara en la revista Nature un algoritmo, basado en redes neuronales, que aprendía por sí mismo a jugar videojuegos de la consola Atari y que superara los puntajes alcanzados por seres humanos en muchos de ellos [21]. Este suceso marcó el primer paso hacia lo que se conoce como una inteligencia artificial general, es decir, una inteligencia que no tiene que estar preprogramada para realizar una tarea y que tiene la capacidad más cercana a lo que entendemos por aprendizaje [22]. Deepmind logró esto por medio de un método de aprendizaje distinto para las redes neuronales que se conoce como aprendizaje por refuerzo (APR), que se diferencia del aprendizaje tradicional, por supervisión, en que

es más apropiado para manejar información que evoluciona en el tiempo. Debido a esto, investigadores de todo el mundo se pusieron a trabajar para extender los resultados de Deepmind a aplicaciones en las que es necesario manejar este tipo de información dinámica, por ejemplo, para los problemas de control [2]. El problema principal existente era que el algoritmo de Deepmind se aplicaba a problemas en las que las acciones que se pueden tomar son siempre finitas, sin embargo, en los problemas de control hay un número infinito de posibles acciones que se pueden tomar. Por ejemplo, en el ajedrez, en cada turno el jugador siempre tiene un número finito de posibles jugadas, mientras que en la tarea de control de posición de un motor DC se tienen infinitas opciones para el voltaje a aplicar debido a la naturaleza física de este. Este problema fue resuelto por Deepmind en 2015 en el cual se propuso el algoritmo de aprendizaje de redes neuronales para problemas en control que ha sido ampliamente usado en el campo de la robótica [9].

A pesar de que se ha mostrado que el aprendizaje por refuerzo es una técnica de control efectiva y prometedora [23], son pocos los casos en los que se reporta aplicar este método en el control de procesos industriales, un campo que tiene requerimientos de control no tan exigentes como los que se encuentran en robótica. Históricamente, el control de procesos industriales se ha caracterizado por adoptar muy lentamente nuevas técnicas de control y prefiere confiar en controladores tradicionales por temas de seguridad y confiabilidad. Por ejemplo, el controlador PID, que apareció en 1911, es la técnica de control más utilizada en el sector industrial y es usado para resolver cerca del 90% problemas de control de procesos existentes en las industrias [24]. Incluso luego del desarrollo de controladores más sofisticados como los controladores adaptivos, inteligentes o predictivos, solo este último ha sido capaz de sustituir al controlador PID en algunas aplicaciones de control de procesos, principalmente en los sectores químicos y petroquímicos [25].

El primer intento de aplicar las técnicas de Deepmind en el control de procesos industriales fue presentado en el año 2017 por investigadores de la Universidad de Alberta en Canadá. Estos estudios se realizaron en aplicaciones de control de humedad en la manufactura de papel y en el de control de columnas de destilación [8]. Sin embargo, los experimentos se realizaron solo en casos muy concretos en los que la dinámica de estos sistemas es lineal y de un orden máximo igual a dos, a pesar que

la gran mayoría de dinámicas encontradas en los procesos industriales es no lineal y de orden mayor a dos. Un segundo intento fue hecho en la unidad de investigación y desarrollo de la compañía japonesa Mitsubishi Electric [26]. Los autores del trabajo aplicaron el aprendizaje por refuerzo para controlar la temperatura de habitaciones por medio de ventiladores industriales, el cual implica trabajar con un sistema muy complejo de controlar ya que la dinámica del sistema tiene que ser descrita por ecuaciones diferenciales parciales. No obstante, las investigaciones solo consideraron el caso en el que los ventiladores solo pueden estar prendidos a una sola potencia o apagados, limitando así el número de posibles acciones a dos, pese a que en la mayoría de aplicaciones industriales el conjunto de acciones posibles es infinito.

En particular, este trabajo está interesado en abordar el control de sistemas de posicionamiento magnético los cuales son usados en muchas industrias. Este tipo de sistemas son empleados cuando las especificaciones de control exigen altos grados de precisión o cuando es indispensable la manipulación de objetos sin contacto mecánico [27]. Por ejemplo, la manufactura de circuitos integrados tiene requerimientos de posición en el orden de los micro y nano metros [28]. Asimismo, los microscopios a escala atómica también requieren posicionamientos con un elevado grado de exactitud [29]. Por otro lado, la industria de la biotecnología requiere la manipulación de células para investigaciones y elaboración de productos [30].

En este sentido, el presente trabajo de investigación propone aplicar el aprendizaje profundo por refuerzo para controlar un sistema de posicionamiento magnético de dos dimensiones. El trabajo se escructura de la siguiente forma. En el primer capítulo, se realiza un repaso a los avances de la inteligencia artificial en los últimos años liderado por las redes neuronales. Asimismo, se presenta la importancia del control de procesos industriales haciendo hincapié en los que requieren de un control de posición. En el segundo capítulo, se presenta el modelo del sistema de posicionamiento magnético que se va a controlar y se estudian algunas técnicas de control relevantes que se aplican en este tipo de problema: control PID, adaptivo e inteligente. En el tercer capítulo, se expone en detalle la propuesta de solución del problema que consiste en una arquitectura actor-critico para el entrenamiento del controlador neuronal. Se presenta el algoritmo Deep Deterministic Policy Gradient (DDPG) que permite el entrenamiento del controlador neuronal. En el cuarto capítulo,

se muestran las simulaciones del sistema de posicionamiento magnético con el controlador neuronal diseñado. Se evalúa el desempeño del controlador frente al seguimiento de varios setpoints y una trayectoria en particular. También se evalúa la capacidad del controlador a rechazar ruido presente en los sensores, perturbaciones externas en la posición del sistema y variación en los parámetros de la planta. Se hacen comparaciones frente a un controlador no lineal y un controlador neuronal. Finalmente, el documento termina presentando las conclusiones de este trabajo, así como las recomendaciones para futuras investigaciones.

Capítulo 1

Problemática

1.1 Historia del aprendizaje por refuerzo

El aprendizaje automático se ha posicionado en los últimos años como una técnica novedosa para automatizar diversas tareas. Esto debido a la aparición de mejoras en los algoritmos relacionados a redes neuronales y al incremento de capacidad computacional con la que se dispone hoy en día [10]. La aparición de las redes neuronales profundas conocidas en inglés como *Deep Neural Networks* ha iniciado una corriente denominada *Deep Learning*, la cual consiste en el uso de redes neuronales con varias capas ocultas como la mostrada en la figura 1.1. A diferencia de las redes neuronales superficiales que solo tienen una capa oculta como la mostrada en la figura 1.2, las redes neuronales profundas pueden extraer mucha más información de los datos de entrada que se le presenten a la red [18]. Por esta razón, este método se ha posicionado como el estado del arte en tareas de reconocimiento de voz, detección de imágenes y procesamiento del lenguaje natural. Por ejemplo, aplicaciones como los asistentes de voz personales como Siri de Apple, Bixby de Samsung, Alexa de Amazon y Cortana de Microsoft son solo algunos ejemplos que utilizan elementos del aprendizaje profundo [31].

Estas técnicas forman parte de lo que se conoce como aprendizaje supervisado, un tipo de aprendizaje que requiere información previamente construida que represente el comportamiento ideal que se desea lograr [18]. Por ejemplo, en la aplicación de detección de objetos, la información que se tendría que proporcionar al algoritmo consistiría en diversos pares (X, Y) en los cuales X corresponde a la imagen de un

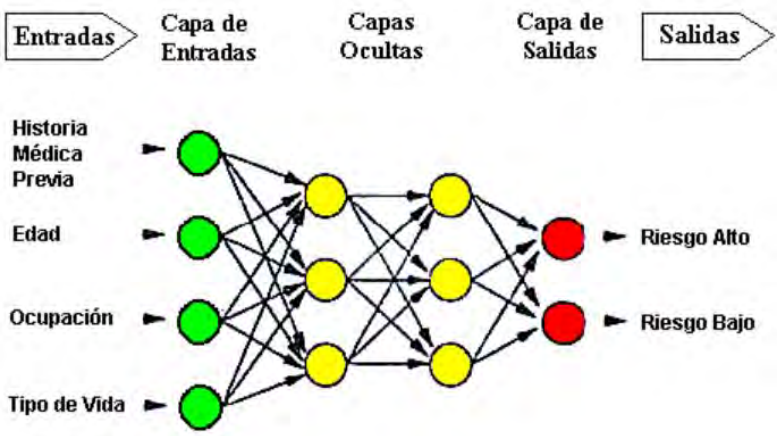

Figura 1.1: Red neuronal profunda. Imagen tomada de `http://perso.wanadoo.es/alimanya/funcion.htm`.

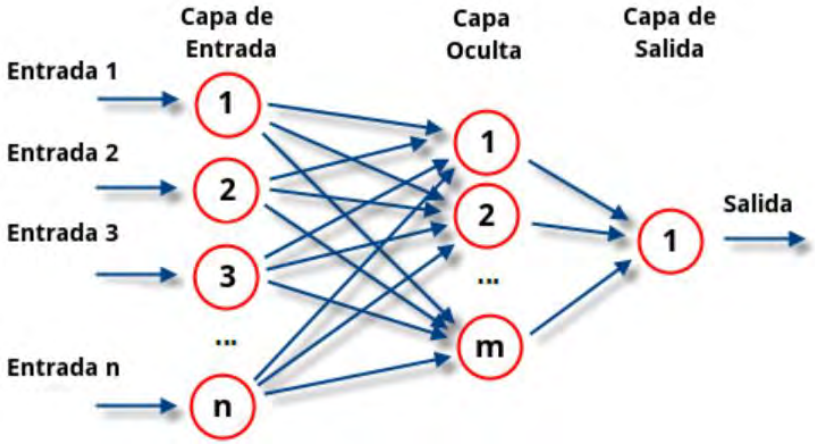

Figura 1.2: Red neuronal superficial. Imagen tomada de `https://es.wikipedia.org/wiki/Perceptron_multicapa`.

objeto e Y al nombre del objeto. En la figura 1.3 se muestra un ejemplo del par de elementos (X, Y), en este caso la imagen del gato formaría parte del conjunto X y la etiqueta "GATO" pertenecería al conjunto Y.

Figura 1.3: Ejemplo de aprendizaje por supervisión en una aplicación de clasificación de imágenes. Imagen tomada de `https://luozm.github.io/cv-tasks`

Sin embargo, el aprendizaje automático también incluye otros tipos de aprendizaje: el aprendizaje por refuerzo y el aprendizaje sin supervisión. En particular, el aprendizaje por refuerzo es un método inspirado en la psicología [13]. El cual consiste en un agente interactuando con un entorno o ambiente, este esquema se representa en la figura 1.4. El agente recibe un premio (puntaje alto) cuando realiza buenas acciones y recibe un castigo (puntaje bajo) cuando realiza lo incorrecto. El problema del aprendizaje por refuerzo es, entonces, el de maximizar el premio que se obtiene en una ventana de tiempo futura. El aprendizaje por refuerzo se ha convertido en los últimos años en una herramiento fundamental para la inteligencia artificial. Con esta se espera producir agentes completamente autónomos que interactúan con su entorno y desarrollan comportamientos óptimos a través de la prueba y error [21].

El desarrollo de estos sistemas inteligentes que pueden responder a su entorno al mismo tiempo que aprenden del resultado de sus acciones es un gran reto a resolver en diversidad aplicaciones. Estas abarcan desde robots, que pueden sensar y actuar con su entorno, hasta sistemas basados puramente en software que interactúan mediante lenguaje natural y archivos multimedia. Así como el aprendizaje profundo ha mejorado la efectividad de muchas aplicaciones, también lo ha hecho en el campo del aprendizaje por refuerzo [13].

La primera aplicación del aprendizaje por refuerzo se remonta a la década de los noventa, cuando Gerald Tesauro del IBM Watson Research Center desarrolló el programa TDGammon [32]. Este consistía en un programa capaz de aprender a jugar Backgammon, que funcionaba en base en una red neuronal, de tres capas, entrenada a

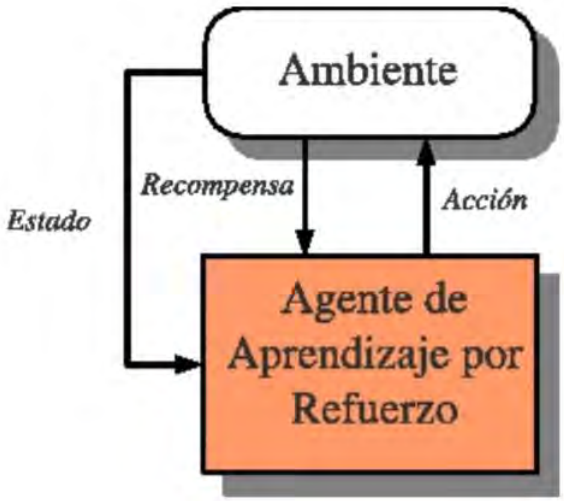

Figura 1.4: Ilustración del proceso de interacción entre un agente y su entorno. Imagen tomada de `http://www.informatiksolutions.com/victor/RL_intro_Victor_Uc.pdf`.

través de lo que se conoce como el aprendizaje temporal-diferencial. El nombre de esta técnica se debe a que se usa la diferencia de la función de costo a optimizar a través de la red neuronal entre dos instantes de tiempo consecutivos en la ley de actualización de los parámetros de la red.

$$w_{t+1} - w_t = \alpha(Y_{t+1} - Y_t) \sum_{k=1}^{t} \lambda^{t-k} \Delta_w Y_k \tag{1.1}$$

En la ecuación 1.1, Y representa la función de costo, w los parámetros de la red neuronal, λ es el factor de aprendizaje, Δ_w la diferencia entre los parámetros de la red en dos iteraciones consecutivas y t representa cada instante de tiempo . Se reporta que este programa llegó a alcanzar un desempeño comparable al de jugadores expertos [32].

Al mismo tiempo, hubo un intento de aplicar aprendizaje por refuerzo en otras áreas como al control de procesos industriales. Por ejemplo, Hoskins y Himmelbau (1991) propusieron un sistema neuronal entrenado por refuerzo para aplicaciones en procesos químicos [33]. Sin embargo, los resultados obtenidos no fueron alentadores

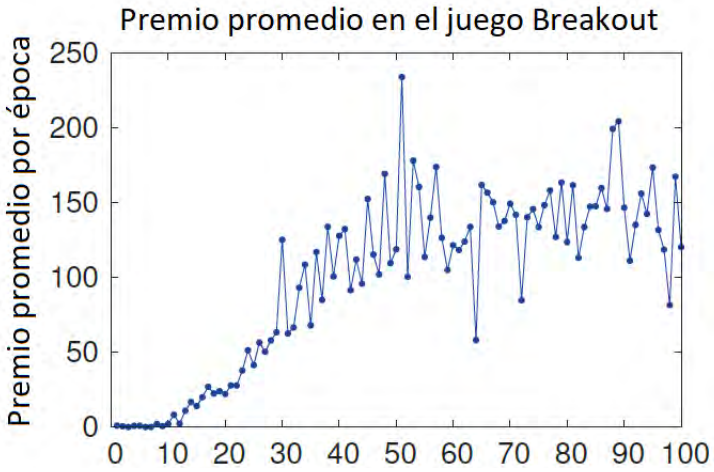

Figura 1.5: Curva de aprendizaje del sistema inteligente creado por Deepmind en el juego Breakout. Se puede ver como el premio que obtiene el agente aumenta con el numero de iteraciones [1].

debido que el desempeño de tal sistema no era superior al controlador PID, además que el diseño y ajuste de este llegaba a ser muy complejo y tedioso.

Desde entonces, el desarrollo del aprendizaje por refuerzo fue más que nada teórico, impulsado por Richard Sutton y Andrew Barto, pero sin muchas aplicaciones prácticas [34]. En parte debido al costo que implica hacer que un sistema alcance el fallo numerosas veces hasta lograr encontrar el comportamiento adecuado y, por otro lado, incluso en simulaciones esta tarea demandaba un poder computacional con el que no se contaba en la época [13].

No fue hasta el año 2015, que la historia cambió completamente para el aprendizaje por refuerzo. Un grupo de científicos de la compañía londinense Deepmind, adquirida por Google en 2014, publicaron en la prestigiosa revista Nature el trabajo titulado *Human-level control through deep reinforcement learning* [21]. En este documento, los autores reportaban que habían sido capaces de diseñar e implementar satisfactoriamente un algoritmo que era capaz de aprender a jugar diversos juegos de la consola Atari, alcanzando en 29 de ellos un desempeño comparable al de un humano experto. La figura 1.5 muestra como luce la curva de aprendizaje del algoritmo de Deepmind en el juego Breakout.

Este sistema consistía en el concepto del aprendizaje Q [34]. El cual es una función que representa qué tan bueno es un estado en un sistema dinámico, respecto al objetivo que se desea alcanzar. El objetivo del aprendizaje Q es encontrar una estimación de esta función de forma iterativa. Una vez encontrada esta estimación, se puede usar

esta para decidir qué acciones tomar dependiendo del estado en el que se encuentre el agente. El principal aporte de Deepmind fue utilizar una red neuronal profunda como estimador de la función Q.

$$Q^*(s,a) = \mathbb{E}[r + \gamma \text{max}_{a'} Q^*(s',a')|, a] \tag{1.2}$$

Donde s es el estado en el que se encuentra el agente, a es la acción que realiza el agente, r es el premio que recibe el agente por realizar la acción a en el estado s, γ es un factor de olvido y $\mathbb{E}$ representa el valor esperado.

Uno de los resultados más impresionantes de este trabajo, es que para el aprendizaje entre los distintos juegos en los que se evaluó el algoritmo, no fue necesario el ajuste de los hiperparámetros del mismo. Los hiperparámetros son los parámetros generales con los que cuenta un algoritmo de inteligencia artificial que se pueden modificar para obtener un desempeño diferente. Usualmente, al aplicar el mismo algoritmo en aplicaciones es necesario cambiar los hiperparámetros; sin embargo, en el sistema propuesto por Deepmind no fue necesario realizar tal ajuste. Tal resultado representa un paso hacia lo que se conoce como inteligencia artificial general, esto es una inteligencia a la que no se le tiene que modificar nada para que pueda aprender a realizar una tarea en cualquier aplicación [22].

Los resultados de Deepmind originaron un creciente interés por el aprendizaje por refuerzo por lo novedoso del método. Esto se refleja en la figura 1.6 que muestra como han ido aumentando el número de publicaciones relacionadas al aprendizaje por refuerzo a lo largo de los años [2]. Sin embargo, había un problema que había que resolver para que el aprendizaje por refuerzo pueda extenderse a aplicaciones concretas como la robótica. La aplicación de Deepmind en videojuegos implicaba un espacio de acción discreto. Esto significa que el numero de acciones a tomar en cierto estado son siempre finitas. En contraste, muchas aplicaciones tentativas del aprendizaje por refuerzo implican un espacio de acción continuo o infinito, como es en el caso de la robótica.

Dicho problema fue resuelto meses después por los mismo investigadores de Deepmind, liderados por Timothy Lillicrap. La solución consistía en aplicar el algoritmo conocido como *Deep Deterministic Policy Gradient* [9] el cual proporciona una nueva forma de calcular el gradiente para la actualización de los parámetros de las

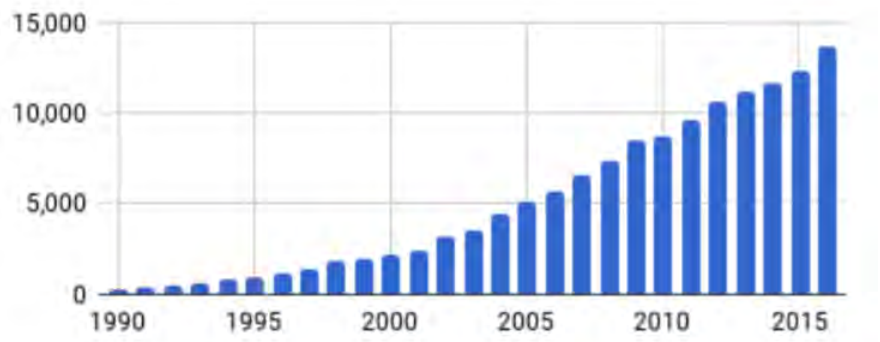

Figura 1.6: Número de trabajos científicos relacionados al aprendizaje por refuerzo a través de los últimos años [2].

redes neuronales.

Luego de esto, el aprendizaje por refuerzo fue adoptado por la comunidad de robótica. Gracias a esta técnica, tareas que antes eran muy complejas de realizar como la manipulación de objetos han sido finalmente posibles de lograr. Por ejemplo, en la figura 1.7 se muestra un brazo robot que es capaz de voltear una tortilla en una sartén [3]. Asimismo, los gigantes de la tecnología que buscan desarrollar los autos que se manejan solos (*self-driving cars*) han invertido fuertemente por el aprendizaje por refuerzo para lograr este cometido [35]. En la figura 1.8 se muestra el nombre de algunas de estas compañías.

Debido a estos sucesos, el presente trabajo pretende investigar la aplicación del aprendizaje por refuerzo en procesos industriales, utilizando como base los recientes avances en el aprendizaje automático como *Q-learning*, *Batch Normalization*, *Experienced Memory Replay* y *Deep Neural Networks*. Específicamente, se buscará controlar un sistema de posicionamiento de dos dimensiones. Para la implementación de esto, se usará el entorno de trabajo Tensorflow de Google.

Figura 1.7: Aplicación del aprendizaje por refuerzo en robótica: Robot que puede voltear una tortilla en una sartén [3].

Figura 1.8: Compañías que buscan desarrollar la tecnología de carros autónomos y que han apostado por el aprendizaje por refuerzo. Imagen tomada de [4].

1.2 Procesos Industriales

1.2.1 Motivación

En el sector industrial son numerosas las aplicaciones que requieren la implementación de sistemas de control. Si bien un gran porcentaje de los problemas industriales se pueden resolver con controladores PID y sus variaciones, hay sectores que requieren el uso de técnicas de control más avanzadas [25]. Por ejemplo, las industrias química y petroquímica han aprovechado el control predictivo basado en modelo. De hecho,

las primeras aplicaciones del control predictivo por modelo como las DMC (Dynamic Matrix Control) y GPC (Generalized Predictive Control) fueron diseñadas para estas industrias [36]. Por otro lado, en robótica es necesario el uso de técnicas avanzadas de control provenientes de la inteligencia artificial como redes neuronales entrenadas por aprendizaje por refuerzo o imitación. Esto debido principalmente a que en robótica los objetivos de control son mucho más complejos de definir. Por ejemplo, el lograr que un brazo robot logre abrir una puerta al girar una manija de manera eficiente y robusta es muy difícil de resolver si el problema se enuncia con objetivos de control de seguimiento a un *setpoint* o trayectoria. Sin embargo, enunciando el problema desde el punto de vista del aprendizaje por refuerzo, este es más sencillo de resolver [10].

1.2.2 Sistemas de Posicionamiento

1.2.2.1 Aspectos generales

El problema de posicionar un objeto en un lugar específico con una alta precisión es común en varios sectores como en la medicina, semiconductores y manufactura de piezas mecánicas. Este tipo de aplicaciones requieren que un objeto se pueda colocar en una posición deseada sin vibración, con gran precisión y en un tiempo adecuado según la aplicación. Además, el sistema de control debe ser capaz de funcionar en un amplio rango de movimiento, en las escalas de los micro-nano metros, y en todos sus grados de libertad [28], [27], [37], [38], [39].

La toma de imágenes a escala atómica es muy importante en el campo de la medicina y biología para el estudio de enfermedades. Los microscopios de efecto túnel son populares por ser efectivos para esta tarea, estos cuentan con sistemas de posicionamiento altamente precisos. Desde su invención en 1980 por Gerd Binnig y Heinrich Rohrer, científicos de IBM en Zurich, el control de posición de estos microscopios ha sido ampliamente estudiado [29].

En la industria de los semiconductores, los sistemas de posicionamiento también son usados para la fabricación de circuitos integrados por medio de litografía [40]. En esta industria, la longitud de canal de un transistor, es un parámetro que determina el tamaño mínimo que se puede obtener en cierta tecnología. El largo de canal de un transistor hace referencia a la distancia entre el surtidor y drenador en un transistor. Actualmente, la mayoría de aplicaciones comerciales en electrónica hacen uso de

tecnologías que permiten transistores con un largo de canal de alrededor de 14 nm; sin embargo, ya se reportan tamaños de canal de 10 nm en algunos dispositivos e incluso longitudes de canal mucho menor en las etapas de investigación en compañías líderes en el sector como Samsung, Qualcomm, ARM, etc [41]. Asimismo, para los transistores con juntura compuesta por metal, óxido y semiconductor, conocidos como transistores MOS se requieren sistemas de posicionamiento que puedan colocar estos materiales en las distancias necesarias para el correcto funcionamiento del transistor [42].

Muchos de estos sistemas de posicionamiento funcionan con actuadores de materiales piezoeléctricos debido a su bajo coste e implementación [38]. Sin embargo, estos presentan fuertes desventajas desde el punto de vista de control. Primero, los materiales piezoeléctricos son muy sensibles a la temperatura, con lo cual la precisión del sistema del control dependerá fuertemente de esta. Segundo, estos actuadores presentan el efecto histéresis, esto es, son incapaces de repetir una misma posición de manera similar. Tercero, estos generan vibración cuando los voltajes aplicados son altos, lo que afecta el error estacionario en ciertos rangos de posición [27]. Por estas razones algunas aplicaciones optan por usar otros tipos de actuadores, como los electromagnéticos [26], [39], [28].

1.2.2.2 Sistemas de Posicionamiento Magnético

Los sistemas de posicionamiento magnéticos eliminan las desventajas de los sistemas con actuadores piezoeléctricos: sensibilidad a la temperatura, histéresis y vibración [39]. Adicionalmente, no están sujetos a otras complicaciones para el control que son propias de sistemas mecánicos como el *backlash* y la fricción seca. Sin embargo, la naturaleza electromagnética de los actuadores trae consigo dos desventajas principales.

Desde el apartado económico, un sistema magnético es más costoso de implementar. Debido a la inexistencia de monopolos magnéticos, un solo actuador no puede mover un objeto en dos sentidos en la misma dirección. Por lo que para lograr el movimiento completo en un grado de libertad es necesario usar dos actuadores [37]. Desde el punto de vista del control, los sistemas magnéticos presentan fuertes no linealidades que son difíciles de tratar por medio de técnicas de linealización simples. Por lo que, las técnicas de linealización populares por su simplicidad, como la lineal-

ización basada en serie de Taylor alrededor de un punto de equilibrio, no son efectivas. Debido a esto, es imprescindible el uso de técnicas de control avanzadas en sistemas de posicionamiento magnético [38].

Durante la última década se han propuesto varias técnicas para el control de estos sistemas. Entre las más resaltantes se pueden encontrar las siguientes. En [39], se propusieron técnicas de control lineales basadas en modelos de parámetros dependientes lineales. El método consistía en la variación de parámetros en un modelo lineal junto con técnicas de control de ganancia programada robusta. En [38], se propuso un método de regímenes deslizantes que usaba métodos de identificación en línea para ajustar continuamente las ganancias en la ley de control. En [28] se trabajó con el enfoque de control adaptivo basado en modelo de referencia usando controladores PID. Finalmente, en [6] y [7] se aplicaron redes neuronales entrenadas por refuerzo y técnicas de linealización.

1.3 Objetivos

1.3.1 Objetivos Generales

- Control de un sistema de posicionamiento magnético de dos dimensiones usando la técnica del aprendizaje profundo por refuerzo.

1.3.2 Objetivos Específicos

- Implementación de un algoritmo de aprendizaje por refuerzo basado en *Deep Deterministic Policy Gradient* para el entrenamiento del controlador.
- Validación del desempeño del controlador frente a diferentes setpoints y una trayectoria.
- Evaluación del desempeño del controlador frente al ruido, perturbaciones externas y cambios en los parámetros de la planta.
- Comparación del controlador propuesto frente a un controlador no lineal y un controlador neuronal.

Capítulo 2

Sistema de posicionamiento magnético de dos dimensiones: Modelado y control

2.1 Modelado de la planta

2.1.1 Descripción de la planta

El sistema de posicionamiento magnético de dos dimensiones con el cual se trabajará es el presentado en [6] y [7], los cuales se basan en [28] y [27]. Este sistema consiste en una rejilla móvil que es posicionada por cuatro electroimanes idénticos. El sistema no presenta un acoplamiento entre los dos ejes de acción de la plataforma. Asimismo, este sistema cuenta con todos los sensores necesarios para la medición de posición, velocidad de la rejilla y de las corrientes en los cuatro electroimanes.

Los electroimanes utilizados en estos sistemas tienen la característica de presentar una fuerza de atracción de la siguiente forma:

$$F_m(i, d_0) = K_f(\frac{i}{d_0})^2 \tag{2.1}$$

Donde i es la corriente del electroimán, d_0 es la distancia entre el electroimán y la rejilla, y K_f es la constante de actuación del electroimán.

En la tabla 2.1 se detallan todos los parámetros que caracterizan al sistema.

Tabla 2.1: Parámetros del sistema de posicionamiento magnético [6].

Parámetro	Descripción	Valor
β	Coeficiente de amortiguamiento de la rejilla	$0.1\frac{\text{N.s}}{\text{m}}$
K_f	Constante de actuación de los electroimanes	$0.49\frac{\text{N.m}^2}{\text{A}^2}$
i_0	Corriente nominal de los electroimanes	3 A
v_0	Voltaje nominal de los electroimanes	18 V
$2d$	Distancia entre dos electroimanes de un mismo eje	50 cm
R	Resistencia de los electroimanes	6Ω
L	Inductancia de los electroimanes	0.001 H
m	Masa movible de prueba	50 g

2.1.2 Derivación del modelo matemático de la planta

El modelo matemático de la planta se obtiene con ayuda de la figura 2.1. Asimismo, hay que separar las componentes mecánica y eléctrica del sistema. La parte eléctrica de la planta esté conformada por los electroimanes. Cada electroimán se modela con el circuito de la figura 2.2, el cual está formado por una resistencia e inductancia conectadas en serie con una fuente de voltaje.

Considerando v el voltaje de alimentación de un electroimán, i la corriente que fluye por el circuito, R la resistencia del electroimán y L la inductancia del electroimán, se puede escribir la ley de voltajes de Kirchoff como:

$$v = Ri + L\frac{di}{dt} \tag{2.2}$$

Por otro lado, la parte mecánica está compuesta por la rejilla metálica, la cual se modela con una masa puntual m que se mueve bajo la acción de las fuerzas electromagnéticas generadas por los electroimanes y por una fuerza de amortiguamiento característica del medio. Adicionalmente, debido a que los ejes X e Y no se encuentran acoplados, cada uno de estos se puede analizar por separado. Aplicando las leyes de Newton a la rejilla metálica se puede escribir para el eje X:

$$m\dot{x} = -\beta\dot{x} - K_f[(\frac{i_1}{d-x})^2 - (\frac{i_2}{d+x})^2] \tag{2.3}$$

Donde i_1 e i_2 son las corrientes de los electroimanes 1 y 2 respectivamente, y x es la posición en el eje X de la rejilla metálica. Asimismo, β es una constante de amortiguación, K_f es la constante de actuación de los electroimanes y d es la distancia entre dos electroimanes en un mismo eje.

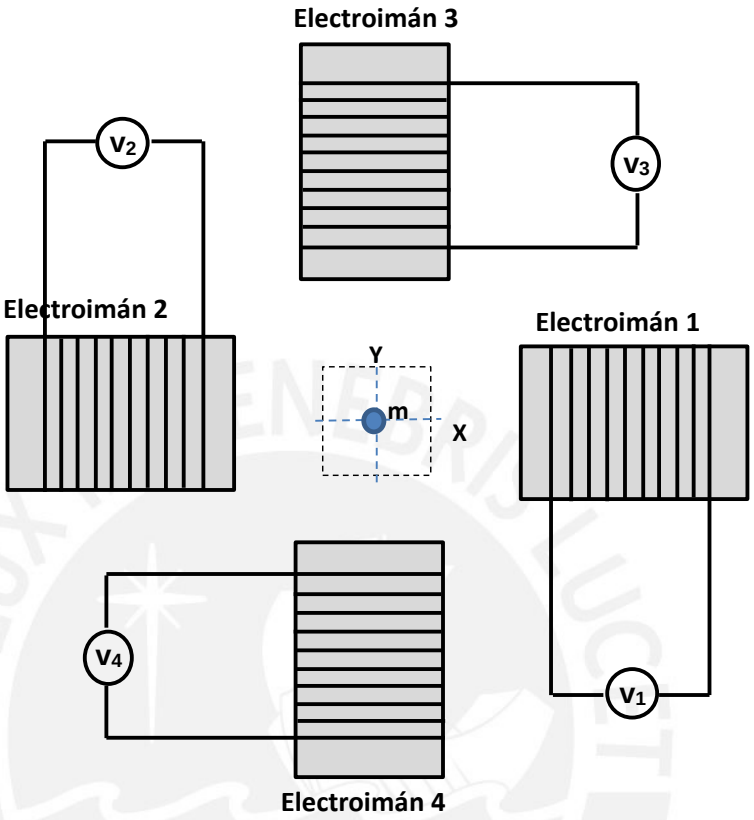

Figura 2.1: Modelo del sistema de posicionamiento magnético de dos dimensiones.

Para simplificar las variables en el modelo se hace uso de los conceptos de voltajes y corriente diferenciales ([40], capítulo 3), los cuales se definen de la siguiente manera:

$$u = v_1 - v_2 \tag{2.4}$$

$$\Delta i = i_1 - i_2 \tag{2.5}$$

Asimismo, se pueden obtener las corrientes y voltajes no diferenciales a partir de sus cantidades diferenciales de la siguiente manera:

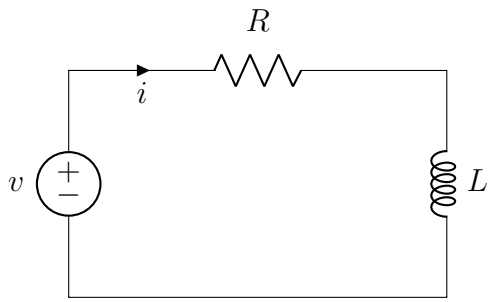

Figura 2.2: Modelo del electroimán.

$$\begin{aligned} i_1 &= i_0 + \frac{x_3}{2} \\ i_2 &= i_0 - \frac{x_3}{2} \end{aligned} \tag{2.6}$$

$$\begin{aligned} v_1 &= v_0 + \frac{u}{2} \\ v_2 &= v_0 - \frac{u}{2} \end{aligned} \tag{2.7}$$

Notar que el procedimiento descrito se puede aplicar al eje Y con los electroimanes 3 y 4. Finalmente, usando estas ecuaciones y extendiendo los resultados al eje Y, se puede formar el modelo espacio estado no lineal de la planta que se muestra en las ecuaciones 2.8 y 2.9.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2.8}$$

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{\beta}{m}x_2 + \frac{K_f}{m}[(\frac{i_0 + \frac{x_3}{2}}{d - x_1})^2 - (\frac{i_0 - \frac{x_3}{2}}{d + x_1})^2] \\ \dot{x}_3 &= -\frac{R}{L}x_3 + \frac{1}{L}u_1 \\ \dot{x}_4 &= x_5 \\ \dot{x}_5 &= -\frac{\beta}{m}x_5 + \frac{K_f}{m}[(\frac{i_0 + \frac{x_6}{2}}{d - x_4})^2 - (\frac{i_0 - \frac{x_6}{2}}{d + x_4})^2] \\ \dot{x}_6 &= -\frac{R}{L}x_6 + \frac{1}{L}u_2 \end{aligned} \tag{2.9}$$

Tabla 2.2: Descripción de las variables del sistema.

Variable	Descripción
x_1	Posición (Eje X)
x_2	Velocidad (Eje X)
x_3	Corriente diferencial en los electroimanes (Eje X)
x_4	Posición (Eje Y)
x_5	Velocidad (Eje Y)
x_6	Corriente diferencial en los electroimanes (Eje Y)
u_1	Voltaje diferencial en los electroimanes (Eje X)
u_2	Voltaje diferencial en los electroimanes (Eje Y)

En la tabla 2.2 se resume el significado de cada variable de estado.

2.2 Técnicas de control para el control de posición de un sistema de posicionamiento magnético

2.2.1 Control PID-MRAC

Esta técnica de control combina dos métodos: control PID y adaptivo basado en modelo (MRAC por sus siglas en inglés). Por un lado, el controlador PID propone aplicar como ley de control $u(t)$ la siguiente expresión en tiempo continuo:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{2.10}$$

Donde $e(t)$ es el error entre la variable de estado a controlar y su valor deseado. Además, K_p, K_i y K_i son parámetros de diseño que hay que elegir. Para obtener la expresión en tiempo discreto se deben aproximar las expresiones para la derivada e integral del error. La principal ventaja de este método es que la ley de control no necesita de un modelo matemático de la planta. Asimismo, la ley de control es fácilmente implementable siendo la única complejidad el elegir el método de aproximación que se use para el cálculo de la derivada e integral del error. Sin embargo, el controlador PID no puede cumplir con los objetivos de control en un amplio rango de operación cuando la planta presenta dinámicas no lineales [28].

Por otro lado, el control adaptivo basado en modelo es una técnica compuesta en tres partes. Primero, se define un modelo de referencia el cual se desea que la planta a controlar siga. Segundo, se define un controlador en función de ciertos parámetros

θ. Finalmente, definiendo una función de Lyapunov adecuada se logra obtener una ley de actualización de los parámetros θ del controlador. En la figura 2.3 se representa este método y se puede observar que el error entre la salida de la planta y el modelo de referencia se usa como entrada del bloque encargado de adaptar los parámetros del controlador.

En consecuencia, el controlador PID-MRAC consiste en utilizar el error entre la salida de la planta y el modelo de referencia de la figura 2.3 para cambiar los parámetros K_p, K_i y K_d del PID. De esta manera se logra que el controlador PID amplíe su rango de operación. En [28] se empleó este método para controlar un sistema de posicionamiento magnético que requería operar en el orden de los micrometros. El controlador cumplió con los objetivos de control en un rango de posiciones mayor al de un controlador PID. Sin embargo, no fue suficiente para cumplir todo el rango de operación requerido.

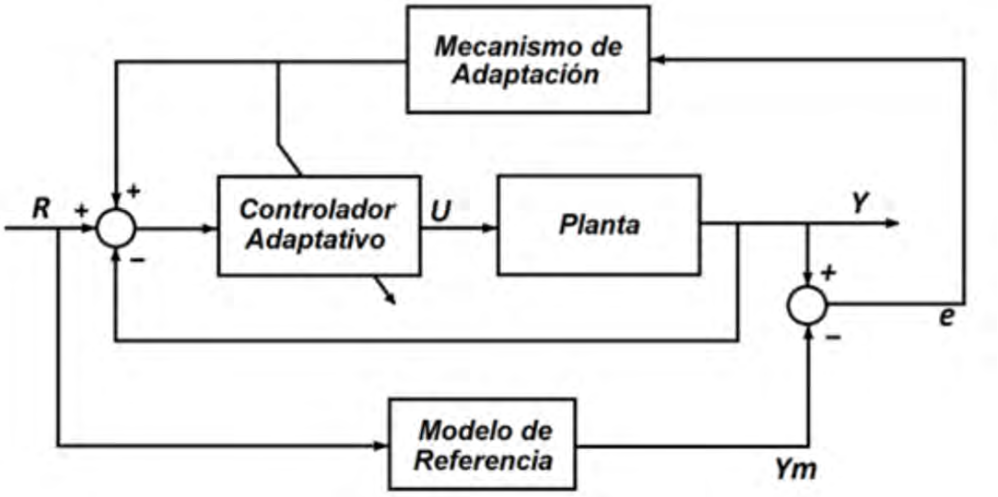

Figura 2.3: Control adaptivo basado en modelo de referencia. Imagen tomada de [5].

2.2.2 Control basado en redes neuronales

2.2.2.1 Nociones básicas

La técnica de inteligencia artificial más usada en control es la basada en redes neuronales unidireccionales o *feedforward* como la mostrada en la figura 2.4. Estas permiten resolver los problemas de modelado y control en sistemas dinámicos. Las redes neuronales son funciones parametrizadas por un vector de parámetros θ , los cuales son tales que buscan optimizar cierta función de costo J que permita resolver el problema de modelado, control o ambos.

La representación matemática de una red neuronal se puede escribir de la siguiente forma:

$$\mathbf{y} = f(\mathbf{x}, \theta) \tag{2.11}$$

Donde el vector $\mathbf{x}$ representa los datos de entrada al sistema, el vector $\mathbf{y}$ es la salida de la red neuronal y θ representa al vector de parámetros de la red neuronal. Notar que el vector x también podría incluir en alguna de sus dimensiones a los datos de salida de la red neuronal, en tal caso se trataría de una red neuronal recursiva como la mostrada en la figura 2.5. En caso no exista esta dependencia la red neuronal es del tipo unidireccional como la mostrado en la figura 2.4.

La búsqueda de los parámetros óptimos θ se conoce como entrenamiento o aprendizaje de la red neuronal y es el principal problema al usar redes neuronales. Como se adelantó en el capítulo anterior los dos tipos de aprendizaje para la red neuronal: supervisado y por refuerzo. A continuación de describen tres enfoques de control basados en redes neuronales.

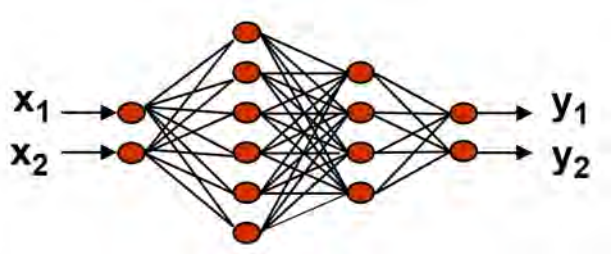

Figura 2.4: Red neuronal conectada completamente.

2.2.2.2 Control usando redes neuronales recursivas

Este es el método más antiguo que se usa con redes neuronales. La técnica utiliza el algoritmo *Dynamic Back Propagation* (DBP), introducido en 1989 por Kumpati Narendra [43] [44], para el aprendizaje de la red neuronal recursiva de la figura 2.5. En este caso la función de costo J a minimizar viene dada por:

$$J = \frac{1}{2}\sum_{k=1}^{k=N}(x_k - \bar{x}_k)^T(x_k - \bar{x}_k) \tag{2.12}$$

Donde $\bar{x}_k$ representa el valor deseado para el vector de estado x_k.

Los parámetros son actualizados por el método del gradiente descendiente:

$$\begin{aligned} v_{ij} &= v_{ij} - \alpha\frac{\bar{\partial J}}{\partial v_{ij}} \\ w_{jk} &= w_{jk} - \alpha\frac{\bar{\partial J}}{\partial w_{jk}} \end{aligned} \tag{2.13}$$

Donde los símbolos $\frac{\bar{\partial J}}{\partial v_{ij}}$ y $\frac{\bar{\partial J}}{\partial w_{jk}}$ hacen referencia a las derivadas parciales totales de J respecto a los parámetros v_{ij} y w_{jk} respectivamente. El algoritmo para el cálculo estas derivadas es el DBP descrito en [44].

En [45] se aplicó esta técnica combinada con un enfoque de lógica difusa para entrenar una red neuronal difusa. Esto significa que una capa de la red neuronal se reserva para implementar las funciones de membresía necesarias en la lógica difusa para calcular la señal de control [46]. El controlador permitía alcanzar diversas posiciones de manera precisa y no requería de un modelo de la planta; sin embargo, el controlador era muy sensible ante cambios en la planta.

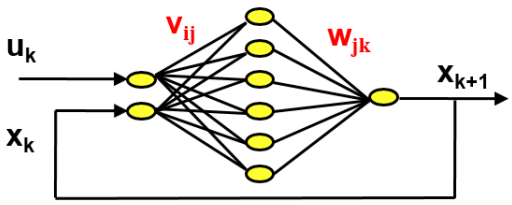

Figura 2.5: Red neuronal recursiva para el control de un sistema dinámico $\mathbf{x_{k+1}} = \mathbf{f}(\mathbf{x_k}, \mathbf{u_k})$.

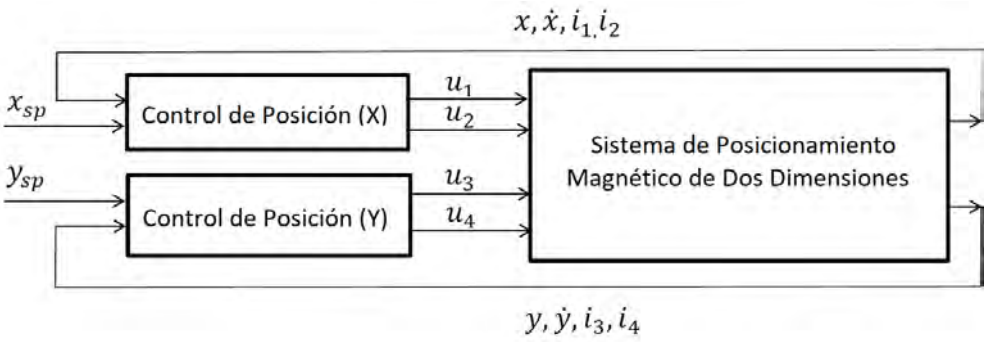

Figura 2.6: Arquitectura de control presentada en [6].

2.2.2.3 Control usando redes neuronales entrenadas por refuerzo

Hasta la fecha son pocos los casos en los que se reporta el uso de redes neuronales entrenadas por refuerzo en aplicaciones industriales. En [8], se hace un primero intento de aplicar los recientes avances en *Deep Learning* al control de procesos industriales. En particular, se estudiaron sistemas de primer orden del tipo SISO (*Single Input Single Output*) y del tipo MIMO (*Multiple Input Multiple Output*) en procesos de manufactura de papel y columnas de destilación. Recientemente, [6] y [7] extienden estos métodos a los sistemas de posicionamiento magnético que se estudian en este trabajo. Por un lado [6], trabaja con dos controladores por separado como se muestra en la figura 2.6, uno para el control de la posición en el eje X y otro para el eje Y. De esta manera, se entrenan dos modelos de orden tres.

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\frac{\beta}{m}x_2 + \frac{K_f}{m}[(\frac{i_0 + \frac{x_3}{2}}{d - x_1})^2 - (\frac{i_0 - \frac{x_3}{2}}{d + x_1})^2] \\
\dot{x}_3 &= -\frac{R}{L}x_3 + \frac{1}{L}u
\end{aligned} \tag{2.14}$$

Luego, se linealiza la dinámica no lineal de la planta por una aproximación simple de Taylor de primer orden para obtener un modelo de la forma:

$$\Delta\dot{\mathbf{x}} = A\Delta\mathbf{x} + B\Delta\mathbf{u} \tag{2.15}$$

Donde:

$$A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \tag{2.16}$$

$$B = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \tag{2.17}$$

Estas matrices toman los siguientes valores en alrededor del punto de equilibrio:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{4K_f}{m} & -\frac{\beta}{m} & \frac{2K_f}{m} \\ 0 & 0 & -\frac{R}{L} \end{bmatrix} \tag{2.18}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} \tag{2.19}$$

Los resultados obtenidos en [6] muestran que esta técnica es capaz de controlar el sistema deseado. Sin embargo, el controlador presentaba un limitado rango de operación y era fuertemente dependiente de los setpoints que se usaban durante el entrenamiento. Por ejemplo, el controlador no era capaz de alcanzar posiciones no vistas durante el entrenamiento.

Por otro lado, en [7] se buscó obtener otro sistema lineal dado por $\dot{\mathbf{z}} = A\mathbf{z} + B\mathbf{w}$ por medio de la linealización por realimentación [47]. Este esquema de control se ilustra en la figura 2.7. Asimismo, se trabajo con un solo controlador, entrenando de esta manera un modelo de orden seis. En este caso las matrices A y B toman los siguientes valores:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.20}$$

Este método permitió aumentar el rango de operación que se alcanzaba con [6].

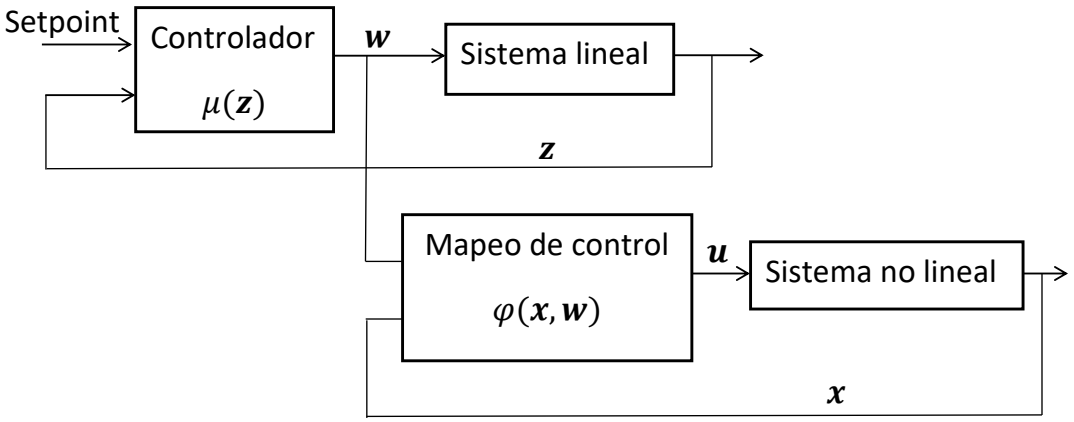

Figura 2.7: Arquitectura de control presentado en [7].

Sin embargo, el número de iteraciones que requería el algoritmo aumentaba y el diseño necesitaba de un modelo exacto del sistema. Asimismo, el entrenamiento del controlador aún usaba datos de la planta linealizada para facilitar el aprendizaje y aún no era capaz de alcanzar setpoints no vistos durante el entrenamiento.

Este capítulo presentó el modelo espacio-estado del sistema de posicionamiento magnético que se desea controlar. Asimismo, se analizaron las ventajas y desventajas de las técnicas que se utilizan para controlar estos sitemas, las cuales se basan en controladores adaptivos, PID y neuronales.

Capítulo 3

Entrenamiento del controlador neuronal

3.1 Definiciones preliminares

El entrenamiento del controlador se basa en el aprendizaje por refuerzo [34]. Dentro de este enfoque, el controlador va mejorando su desempeño conforme va interactuando con la planta mediante prueba y error. A cada intento del aprendizaje por refuerzo del controlador se le conoce como época. En cada época, el controlador interactúa con la planta, que se encuentra en el estado s_t, mediante una acción a_t y por cada interacción recibe un premio r_t, el cual es una función escalar que tiene como entradas el estado en el que se encuentra la planta s_t y la acción que ejecuta el controlador a_t. Estas interacciones se pueden apreciar en la figura 3.1. La función de premio se define de manera específica para cada problema de manera que refleje los objetivos de control que se desean alcanzar [21]. Es importante notar que la acción a_t que ejecuta el controlador se obtiene mediante $a_t = \mu(s_t)$, donde μ representa la política de control. Como en cada época el controlador interactúa varias veces con la planta, es conveniente definir el premio total acumulado en esa época. En particular, la forma más usual de definir al premio total acumulado en una época es de la siguiente forma:

$$R_t = r_t + \gamma r_{t+1} + ... + \gamma^T r_{t+T} \tag{3.1}$$

Donde r_t representa el premio que recibe el controlador al ejecutar la acción de

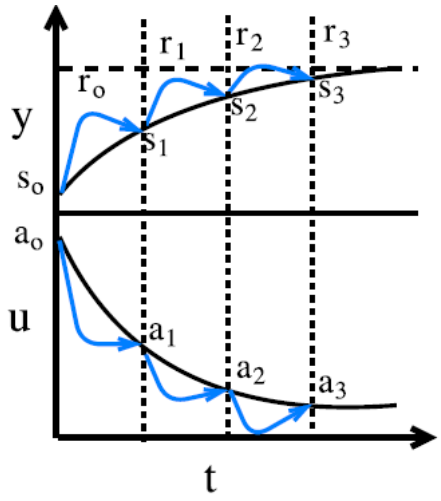

Figura 3.1: Premios [8].

control a_t cuando la planta se encuentra en el estado s_t y γ es un factor de olvido que toma valores entre 0 y 1.

Adicionalmente, en el aprendizaje por refuerzo es necesario definir una función que mida qué tan buena decisión es tomar la acción de control a_t cuando la planta está en el estado s_t. A esta función se le conoce como la función Q y es la columna vertebral de la técnica del entrenamiento por refuerzo. Esta se define de la siguiente forma:

$$Q(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t] \tag{3.2}$$

Esta función representa el premio total acumulado que se espera obtener al aplicar la acción a_t cuando la planta se encuentra en el estado s_t. De esta forma, el objetivo principal del aprendizaje por refuerzo se puede enunciar como la búsqueda de la ley, o política, de control μ que genera las acciones que maximizan la función Q.

$$Q^*(s_t, a_t) = \max_\mu \mathbb{E}[R_t|s_t, a_t] \tag{3.3}$$

El encontrar la función óptima Q es importante porque la acción a elegir en cada instante de tiempo sería aquella que produzca Q^*, esto es $a_t^* = \operatorname{argmax}_{a_t} Q^*(s_t, a_t)$. Sin embargo, se presentan dos problemas fundamentales. Primero, el dar con la función Q es, en general, un problema difícil de resolver. Segundo, en cada instante

de tiempo se tiene que resolver un problema de optimización no convexo.

La primera de estas complicaciones se resuelve de la siguiente manera. Para la forma particular que se ha definido la función de costo total acumulada en la ecuación 3.1, se ha demostrado que se puede tener una expresión recursiva, conocida como la ecuación de Bellman [9], para el cálculo de Q^*. Esta viene dada de la siguiente manera:

$$Q^*(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma Q^*(s_{t+1}, a_{t+1})] \tag{3.4}$$

La segunda de las complicaciones se resolverá con el método del actor y crítico en la siguiente sección.

3.2 Método del actor y crítico

Este método de entrenamiento propone usar dos redes neuronales que representan a un actor, que ejecuta la acción, y un crítico, que evalúa qué tan buena es la acción que realiza el actor [34]. Los parámetros de la red neuronal del actor se representan por θ_a y los de la red del crítico por θ_c. El actor es una red neuronal que representa el comportamiento μ_t del controlador y el crítico es otra red neuronal que busca estimar $Q(s_t, a_t)$. Asimismo, en este enfoque las dos redes neuronales aprenden al mismo tiempo. El funcionamiento de esta arquitectura se ve en la figura 3.2. En cada instante de entrenamiento en el que hay que tomar una decisión de control, el actor propone una acción y el crítico evalúa qué tan buena es esta acción. Luego, se realiza una etapa del algoritmo de propagación de errores para estimar los gradientes de μ y Q respecto a sus respectivos parámetros. Finalmente, los parámetros de las redes neuronales se actualizan por medio del método del gradiente descendiente. En [48] se pueden encontrar más detalles teóricos sobre este método.

La siguiente ecuación muestra la función de costo que busca minimizar el crítico:

$$J(\theta_c) = \mathbb{E}[(r_t + \gamma Q'(s_{t+1}, a_{t+1}) - Q(s_t, a_t, \theta_c))^2] \tag{3.5}$$

Donde $r_t + \gamma Q'(s_{t+1}, a_{t+1})$ representa el valor real de la función Q estimado por la ecuación de Bellman 3.4 y $Q(s_t, a_t, \theta_c)$ es el valor dado por la red neuronal para Q. Notar que se usa la notación Q' para referirse a los valores de la función Q en los

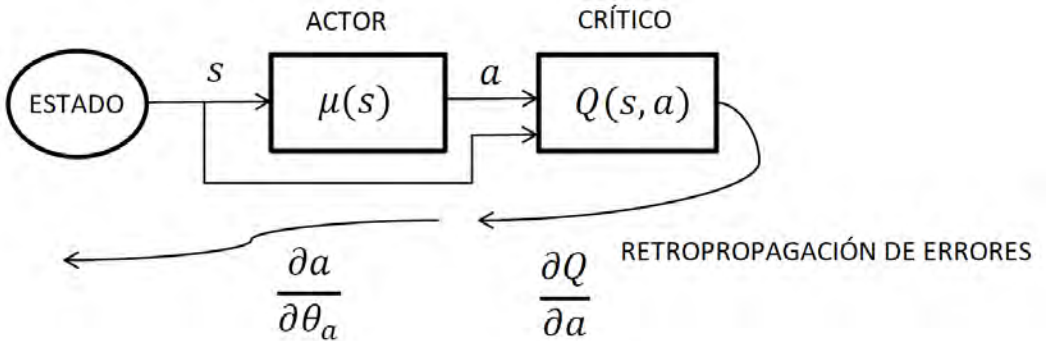

Figura 3.2: Arquitectura actor-crítico.

instantes $t+1$, como se verá en la siguiente sección se usarán dos redes neuronales adicionales para este cometido. El gradiente de esta expresión se obtiene derivando la ecuación 3.5:

$$\frac{\partial J(\theta_c)}{\partial \theta_c} = \mathbb{E}[(r_t + \gamma Q'(s_{t+1}, a_{t+1}) - Q(s_t, a_t, \theta_c))\frac{\partial Q}{\partial \theta_c}] \tag{3.6}$$

Por otro lado, el objetivo del actor es simplemente maximizar el valor Q. Por tanto, para usar el método del gradiente, solo es necesario calcular la siguiente derivada:

$$\frac{\partial Q(s_t, a_t, \theta_a)}{\partial \theta_a} = \mathbb{E}[\frac{\partial Q(s_t, a_t, \theta_a)}{\partial a_t}\frac{\partial a_t}{\partial \theta_a}] \tag{3.7}$$

3.3 Algoritmo DDPG

El algoritmo de entrenamiento a usar es el *Deep Deterministic Policy Gradient* (DDPG) [9]. Este es un algoritmo desarrollado para implementar la estructura actor-crítico del aprendizaje por refuerzo descrito en la sección anterior. En la figura 3.3 se representa de manera simplificada el funcionamiento de este algoritmo. El algoritmo 3.1 es el usado para el entrenamiento del controlador.

Este algoritmo usa los siguientes elementos:

- **Redes objetivo:** Para el cálculo del gradiente en la ecuación 3.6 de la función de costo de la red neuronal del crítico es necesario conocer el valor de $Q'(s_{t+1}, a_{t+1})$ y, a su vez, de $a_{t+1} = \mu(s_{t+1})$. Por esta razón, es conveniente tener dos

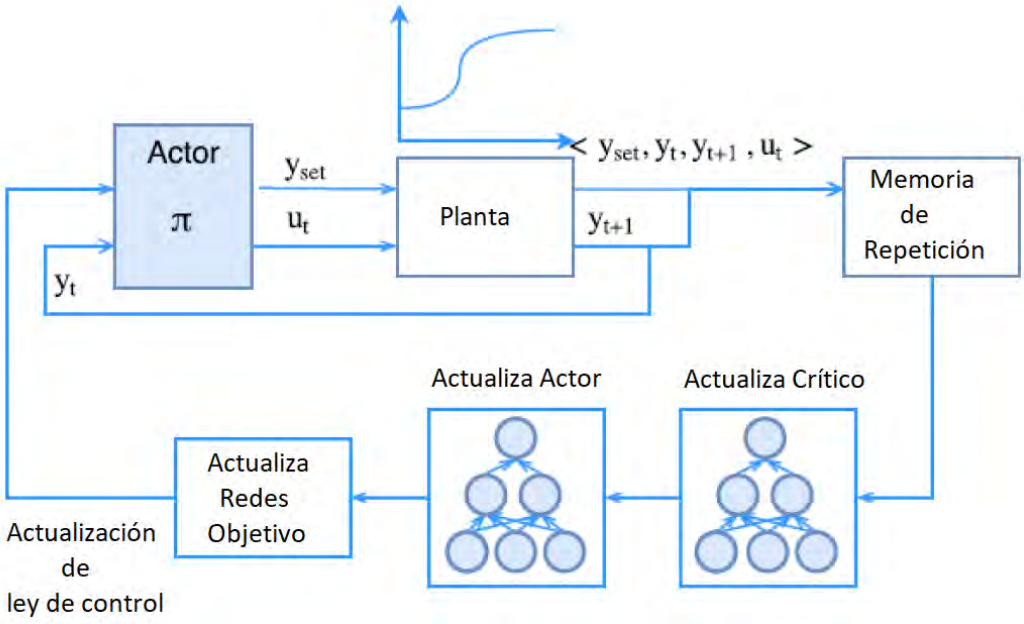

Figura 3.3: Esquema de entrenamiento actor-crítico [8]. Notar que $u_t = a_t$.

redes neuronales adicionales para la estimación de estos dos valores [1]. Los parámetros de estas redes neuronales se representan como : θ'_c para el crítico objetivo y θ'_a para el actor objetivo. Estos se actualizan de la siguiente forma:

$$\begin{aligned} \theta'_c &\leftarrow \tau\theta_c + (1-\tau)\theta'_c \\ \theta'_a &\leftarrow \tau\theta_a + (1-\tau)\theta'_a \end{aligned} \tag{3.8}$$

- **Memoria de repetición:** Para el cálculo de los gradientes 3.6 y 3.7 es necesario realizar la estimación del valor esperado de estas expresiones. Para lograr esto, el algoritmo guarda en cada instante de tiempo cuatro elementos: el estado s_t, la acción a ejecutar a_t, el nuevo estado s_{t+1} generado al aplicar a_t y el premio recibido r_t. A todo este bloque que se almacena se le denomina memoria de repetición. Para la estimación de las expresiones que requieren de una esperanza matemática simplemente se toma el promedio de la expresión que se desea

Algoritmo 3.1 Algoritmo de entrenamiento del controlador neuronal basado en [9] y [8]

Inicializar aleatoriamente θ_a, θ_c, θ'_a, θ'_c
Inicializar la memoria de repetición (MR)
for época = 1, M **do**
    Inicializar el proceso estocástico $\mathcal{N}$ de Ornstein-Uhlenbeck
    Seleccionar aleatoriamente un setpoint y_{sp}
    for instantes = 1, T **do**
        Observar el estado s_t
        Calcular la acción de control $u_t = \mu(s_t, \theta_a) + \mathcal{N}$
        Aplicar la acción de control u_t
        Observar el nuevo estado s_{t+1}
        Calcular el premio $r = r(s_t, a_t, s_{t+1})$
        Almacenar las transiciones (s_t, s_{t+1}, u_t, r) en la MR
        Elegir aleatoriamente n muestras de transiciones de la MR
        Calcular $Q_i = r_i + \gamma Q'_i \; \forall i \in$ muestra de la MR
        Calcular el gradiente $\frac{\partial J(\theta_c)}{\partial \theta_c} = \frac{1}{n}\sum_i^n (Q_i - Q(s^i, a^i, \theta_c)\frac{\partial Q(s^i, a^i, \theta_c)}{\partial \theta_c})$
        Actualizar al crítico: $\theta_c \leftarrow \theta_c + \alpha \frac{\partial J(\theta_c)}{\partial \theta_c}$
        Calcular el gradiente $\frac{\partial Q(s_t, a_t, \theta_a)}{\partial \theta_a} = \frac{1}{n}\sum_i^n \frac{\partial Q(s_t, a_t, \theta_a)}{\partial a_t}\frac{\partial a_t}{\partial \theta_a}$
        Actualizar al actor: $\theta_a \leftarrow \theta_a + \alpha \frac{\partial Q(s_t, a_t, \theta_a)}{\partial \theta_a}$
        Actualizar al crítico objetivo $\theta'_c \leftarrow \tau\theta_c + (1-\tau)\theta'_c$
        Actualizar al actor objetivo $\theta'_a \leftarrow \tau\theta_a + (1-\tau)\theta'_a$
    end for
end for

calcular de una muestra aleatoria de esta memoria de repetición [1].

- **Normalización por lotes:** Se aplica esta técnica presentada en [49] para aumentar la velocidad de entrenamiento de las redes neuronales que consiste en dos cosas. Primero, la salida de cada capa de la red neuronal es normalizada tal que esta tenga un valor promedio igual a cero y una varianza igual a uno. Segundo, los gradientes no se calculan en cada época del algoritmo, sino que se toma un promedio cada N épocas. Donde N es un valor entero y potencia de dos.

- **Gradiente máximo:** El gradiente de la ecuación 3.6 puede hacer que se produzcan valores de la señal de control fuera de los límites físicos permitidos. Por ejemplo, este gradiente puede ocasionar que el voltaje de entrada a aplicar sea mayor al que se puede generar. En particular, se muestra que la siguiente

forma de limitar este gradiente es efectiva para la arquitectura actor-crítico [50].

$$\Delta(p) = \begin{cases} (p_{max} - p)/(p_{max} - p_{min}), \text{si} \Delta(p) > 0 \\ (p - p_{min})/(p_{max} - p_{min}), \qquad \text{en otro caso.} \end{cases} \tag{3.9}$$

Donde p_{min} y p_{max} representan los valores mínimos y máximos de la señal de control.

- **Ruido exploratorio:** En el aprendizaje por refuerzo es necesario introducir ruido $\mathcal{N}$ a las acciones del agente o controlador para que se puedan explorar distintos estados a los propuestos por la política o ley de control μ que puedan llevar a la solución óptima del problema de control. Por ejemplo, al carecer de ruido exploratorio el controlador podría converger en una solución subóptima. Sin embargo, si se explora mucho el algoritmo podría tardar demasiado en converger o no hacerlo nunca. A esto se le conoce como el dilema de la exploración y explotación [34]. En [9] se muestra que el ruido extraído del proceso estocástico de Ornstein-Uhlenbeck [51] es adecuado para varios problemas de control. Asimismo, los resultados obtenidos en [6] y [7] verifican que este ruido permite una buena exploración del espacio-estado de la planta que se estudia en este trabajo. Este ruido se usa de la forma mostrada en el Algoritmo 3.1.

- **Función premio:** En este trabajo se propone el uso de la siguiente función premio:

$$r(s_t, a_t, s_{t+1}) = \begin{cases} c, & \text{si} \mid x^i - x^i_{sp} \mid \leq \epsilon, \forall i \\ -\|\mathbf{x} - \mathbf{x}_{sp}\|_1, & \text{en otro caso} \end{cases} \tag{3.10}$$

Donde $\mathbf{x}$ es el estado de la planta y $\mathbf{x}_{sp}$ el estado deseado de la planta. Asimismo, c y ϵ son parámetros de diseño que hay que buscar. Esta función se elige porque refleja el objetivo de control que se busca conseguir (control a un setpoint). Por ejemplo, si el controlador logra que el estado de la planta se acerque a su estado deseado dentro de una distancia ϵ, este recibiría un premio positivo igual a c. Si la acción no logra acercar a la planta a su estado deseado, el controlador es

penalizado con un puntaje negativo igual a la distancia que lo aleja del estado deseado.

- **Setpoint:** En este trabajo se define la posición final deseada del sistema dinámico a controlar como:

$$\mathbf{x}_{sp} = \begin{bmatrix} x^* \\ 0 \\ \frac{u_1}{R} \\ y^* \\ 0 \\ \frac{u_2}{R} \end{bmatrix} \tag{3.11}$$

Donde x^* e y^* corresponden a las coordenadas finales deseadas (x^*, y^*) en el plano XY del sistema de posicionamiento magnético.

3.4 Estrategia de control

Finalmente, el algoritmo de control se muestra a continuación:

Algoritmo 3.2 Algoritmo de control

1: **function** CONTROLPOSICION(x_1, x_2, i_1, i_2, $x_{1_{sp}}$, x_4, x_5, i_3, i_4, $x_{4_{sp}}$)
2: Calcular $x_3 = i_1 - i_2$ and $x_6 = i_3 - i_4$
3: Construir estado $s_t = [x_1\, x_2\, x_3\, x_4\, x_5\, x_6\, x_{1_{sp}}\, x_{4_{sp}}]^T$
4: Calcular (u_1, u_2) = $\mu(s_t, \theta_a)$
5: Calcular $v_1 = v_0 + \frac{u_1}{2}$ and $v_2 = v_0 - \frac{u_1}{2}$
6: Calcular $v_3 = v_0 + \frac{u_2}{2}$ and $v_4 = v_0 - \frac{u_2}{2}$
7: **return** v_1, v_2, v_3, v_4
8: **end function**

Este capítulo presentó los fundamentos del aprendizaje por refuerzo y arquitecuras actor-crítico utilizadas para resolver el problema de control . También se analizó el algoritmo DDPG y se presentó el diseño de la función de premio que se propone utilizar para el entrenamiento del neurocontrolador.

Capítulo 4

Pruebas de simulación y discusión

4.1 Entrenamiento del neurocontrolador

La arquitectura de las redes neuronales implementadas consta de dos capas ocultas: la primera con 400 neuronas y la segunda con 300. Las funciones de activación de las neuronas son del tipo ReLU [52], $f(x) = \max(0, x)$, para las capas ocultas, mientras que para la capa de salida, la activación es por medio de la función arcotangente. El neurocontrolador fue entrenado por el método del actor-crítico descrito en el capítulo tres. Para el entrenamiento del controlador se usaron valores enteros de setpoints en el rango de -15 cm y 15 cm en ambos ejes. Estos valores cubren el rango completo de acción del sistema de posicionamiento magnético descrito en [6]. En las figuras 4.1 y 4.2 se muestran las curvas de aprendizaje del neurocontrolador obtenidas. Estas curvas tienen como eje horizontal el número de iteración del algoritmo de aprendizaje. El eje vertical es el valor del premio total acumulado de la ecuación 3.1 el cual fue descrito en el capítulo tres. La curva mostrada en la figura 4.1 corresponde al caso en que se considera como setpoint únicamente a la variable de estado de posición [8], esto es, $\mathbf{x} = [x^* y^*]^T$. Esta curva converge aproximadamente en la iteración número 200 del algoritmo DDPG con un valor final de premio total de aproximadamente $17,000$. Por otro lado, la curva mostrada en la figura 4.2 corresponde al caso en el que el setpoint se define como en la ecuación 3.11. En este caso, el algoritmo converge en la iteración número 100 con un valor de premio total acumulado total de aproximadamente $23,000$.

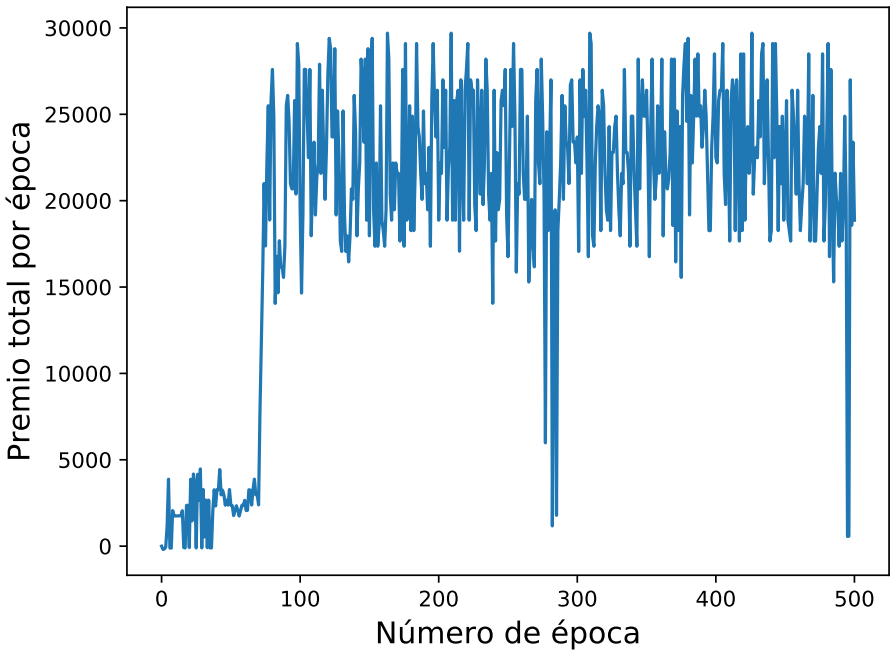

Figura 4.1: Curva de aprendizaje del controlador neuronal con la estrategia propuesta en este trabajo.

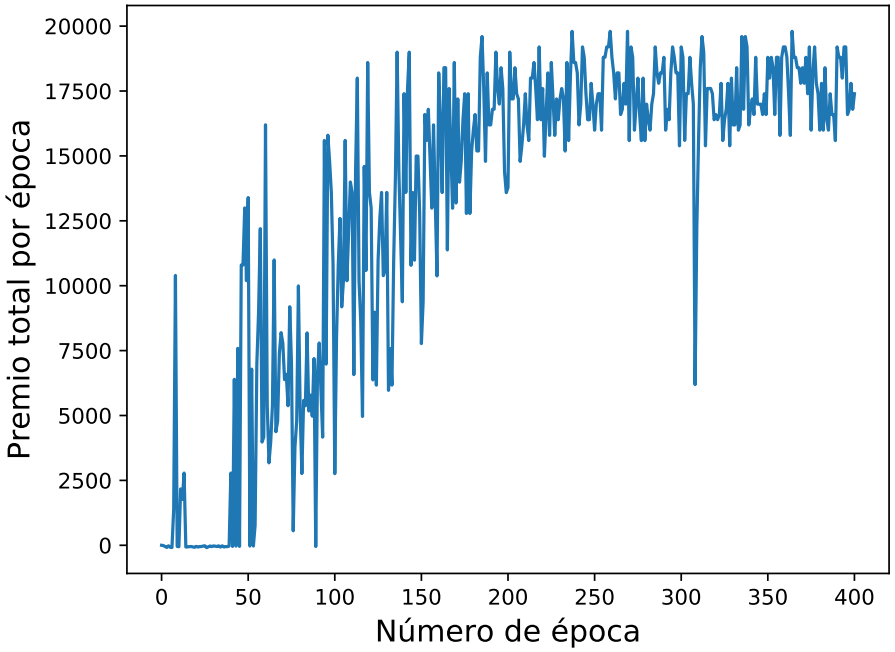

Figura 4.2: Curva de aprendizaje del neurocontrolador neuronal usando lo propuesto en [8].

4.2 Control

4.2.1 Seguimiento de setpoint

En las figuras 4.3, 4.4 y 4.5 se muestra el desempeño del controlador entrenado frente a varios setpoints mostrados y no mostrados durante el entrenamiento. Las figuras 4.6, 4.7 y 4.8 muestran el valor de la señal de entrada de voltaje requerida en los electroimanes para alcanzar los setpoints. En estos casos, se observó valores máximos iguales a 6% y 2% de sobreimpulso y error en estado estacionario respectivamente. Asimismo, se requirieron valores de voltaje entre 10V y 26V para estos casos. Los valores necesarios de corriente estuvieron entre 1.7A y 4.3A. Se verifica que el controlador diseñado es capaz de controlar la posición del sistema incluso a valores de setpoint no mostrados durante el entrenamiento.

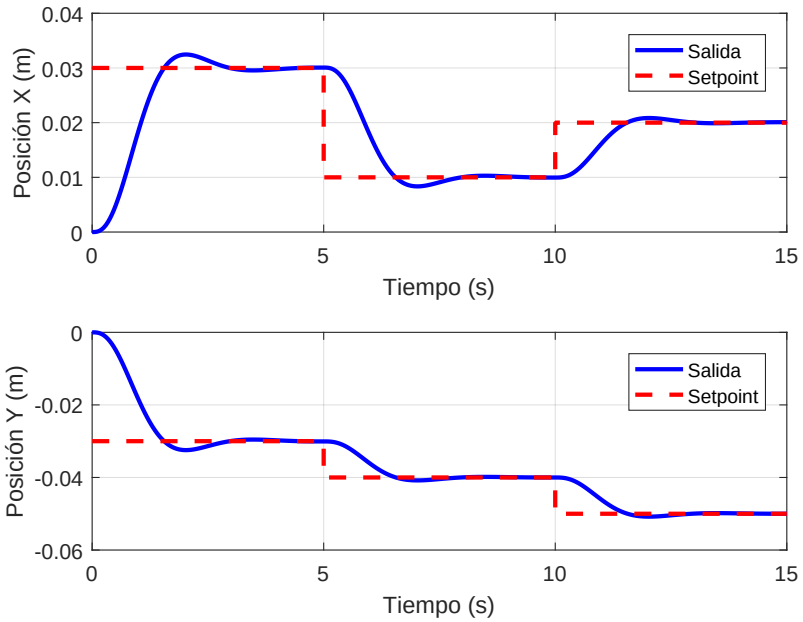

Figura 4.3: Desempeño del controlador con setpoints mostrados en el entrenamiento.

4.2.2 Seguimiento de trayectoria

En las figuras 4.10 y 4.12 se muestra el seguimiento de una trayectoria deseada para los ejes X-Y. De esta manera, se alcanza la forma mostrada en la figura 4.9. Las figuras

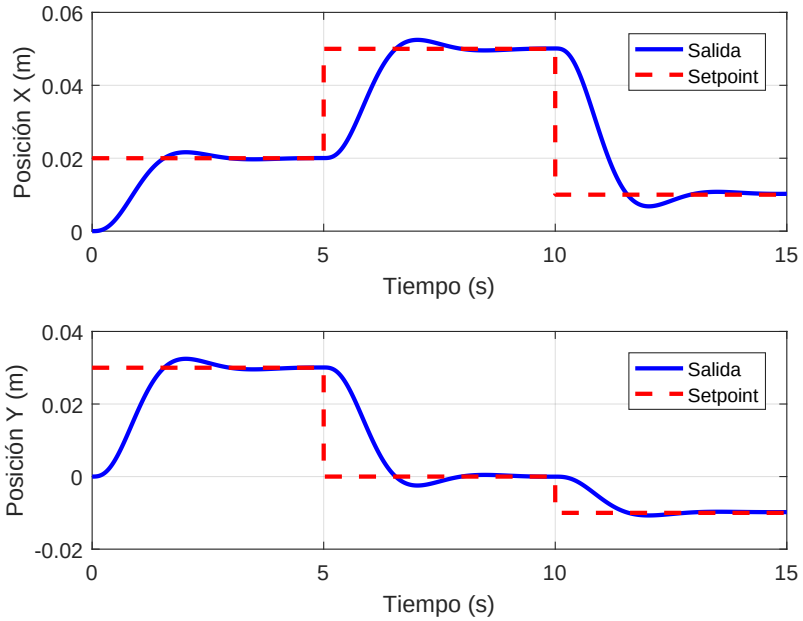

Figura 4.4: Desempeño del controlador con setpoints mostrados en el entrenamiento.

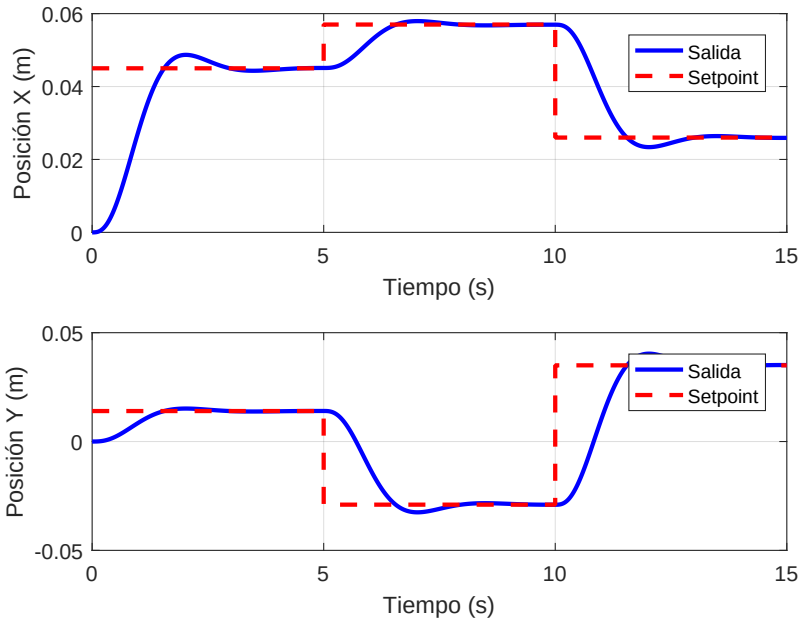

Figura 4.5: Desempeño del controlador con setpoints no mostrados en el entrenamiento.

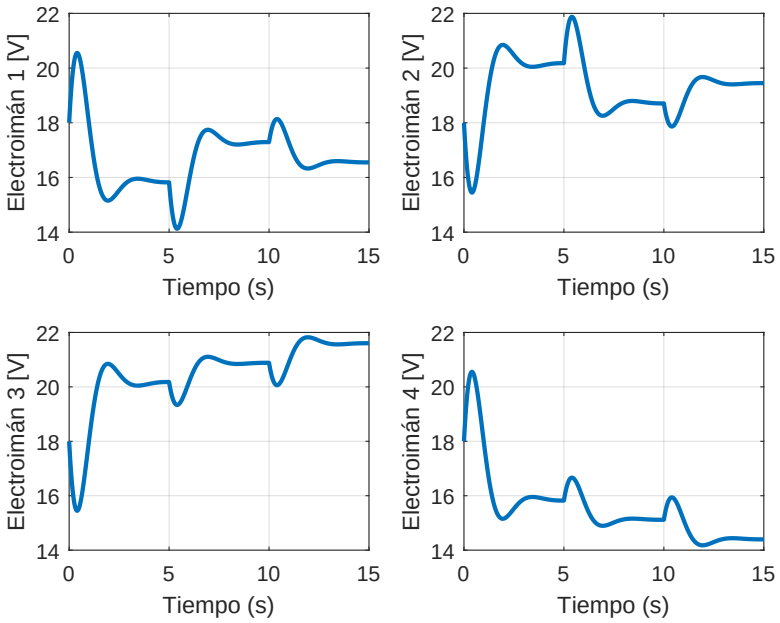

Figura 4.6: Voltajes de entrada para el caso de la figura 4.3.

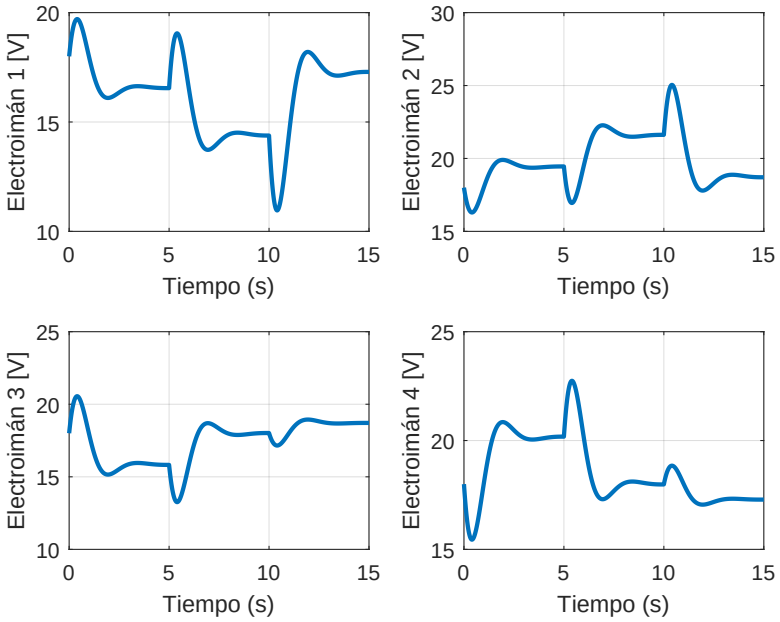

Figura 4.7: Voltajes de entrada para el caso de la figura 4.4.

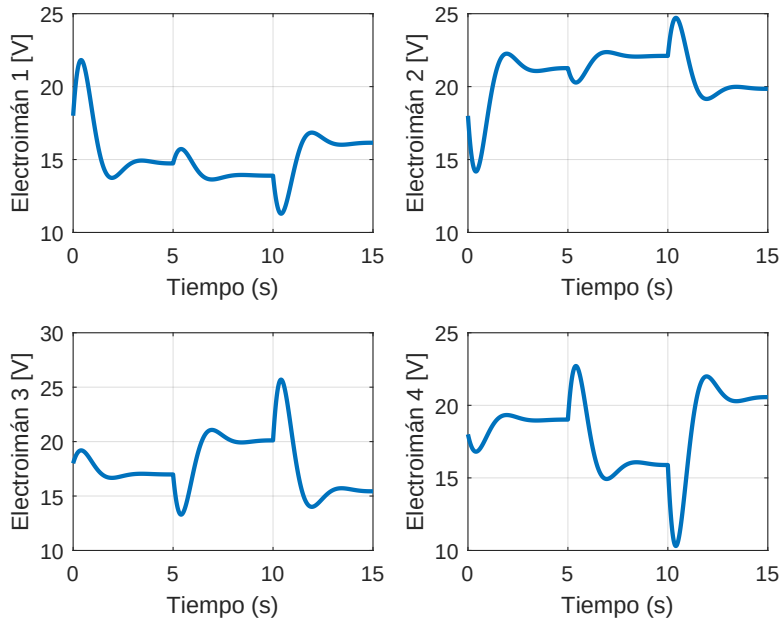

Figura 4.8: Voltajes de entrada para el caso de la figura 4.5.

4.11 y 4.13 muestran los valores requeridos de voltaje en los electroimanes para seguir estas trayectorias. Se observaron valores de voltaje entre 7V y 28V para la generación de esta figura. Los valores de corriente estuvieron entre 1.2A y 4.7A.

4.2.3 Rechazo al ruido

En las figura 4.12 se muestra el desempeño del neurocontrolador frente a un ruido blanco gaussiano aditivo con $\mu = 0$ y $\sigma^2 = 0.1$ en el sensor de posición. Este fue el caso máximo para tener errores menores al 5%. La figura 4.13 muestra los voltajes requeridos en los electroimanes para este escenario.

4.2.4 Rechazo a perturbaciones

El desempeño del controlador diseñado frente a una perturbación se muestra en la figura 4.14. Por simplicidad, se realiza el experimento solo en el eje X. En este caso, la perturbación viene dada por un cambio de posición súbito en el instante $t = 5$s de 5cm a 10cm. Los voltajes correspondientes a este caso se muestran en la figura 4.15.

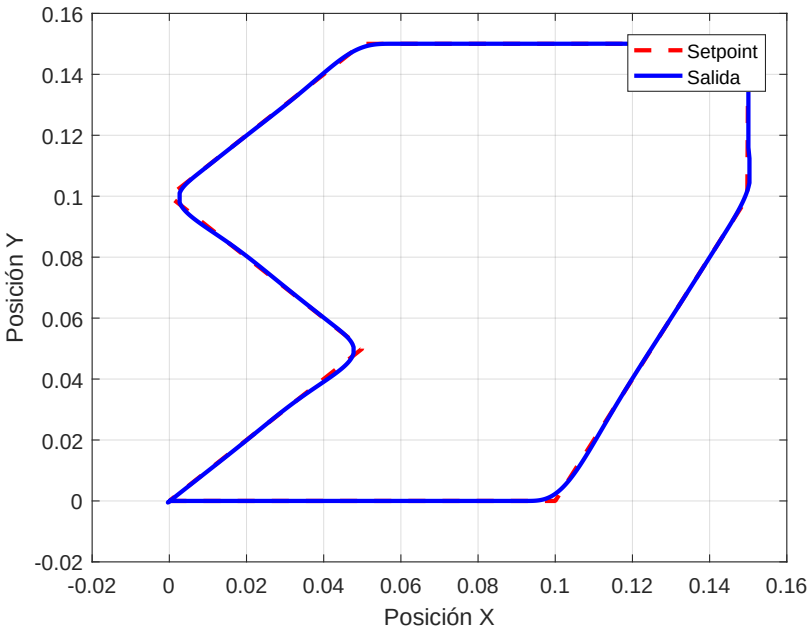

Figura 4.9: Desempeño del controlador frente al seguimiento de una trayectoria.

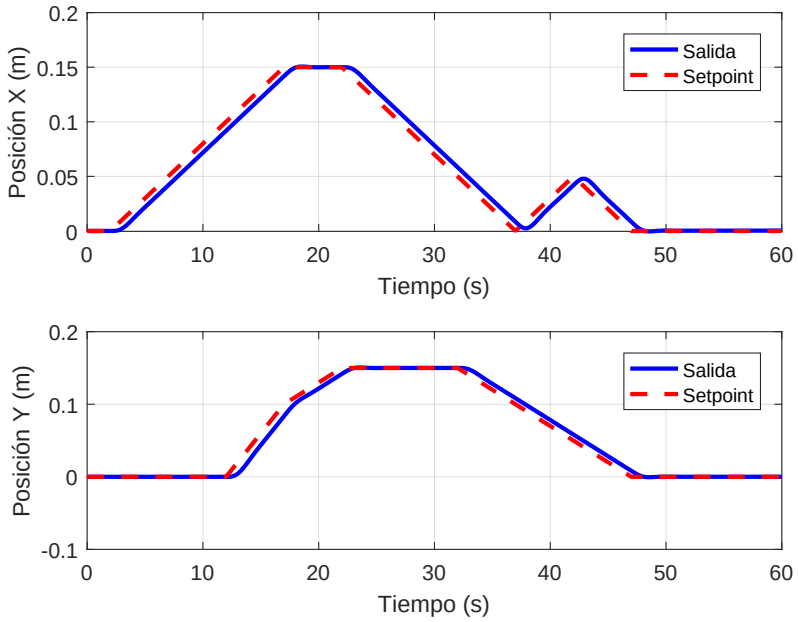

Figura 4.10: Trayectoria en el tiempo para el caso de la figura 4.9.

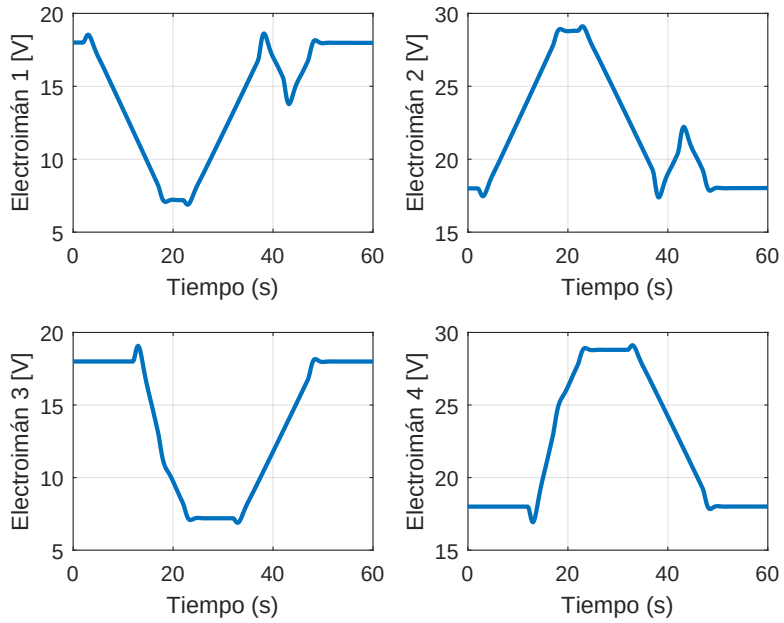

Figura 4.11: Voltajes de entrada para obtener la trayectoria de la figura 4.9.

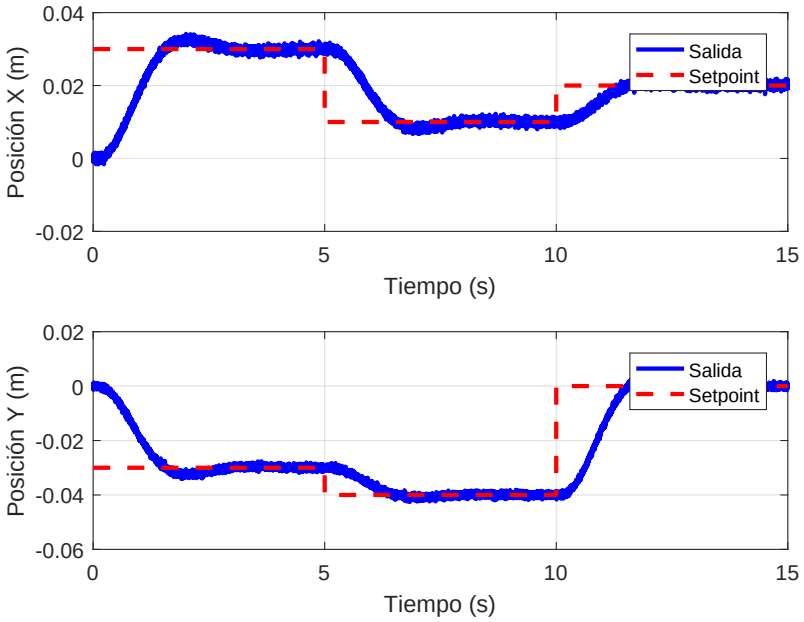

Figura 4.12: Desempeño del controlador frente al ruido.

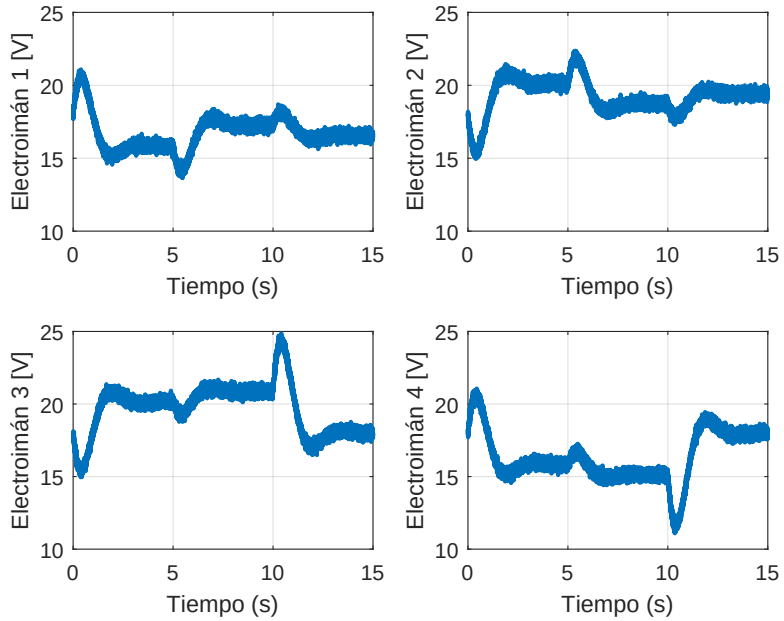

Figura 4.13: Voltajes de entrada para el caso de la figura 4.12.

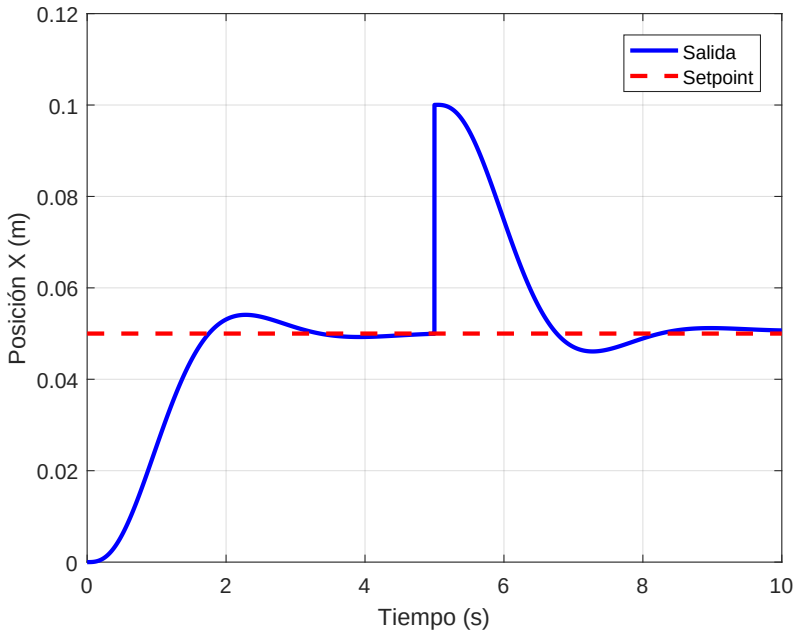

Figura 4.14: Desempeño del controlador frente a una perturbación de 5cm en el instante $t = 5$s.

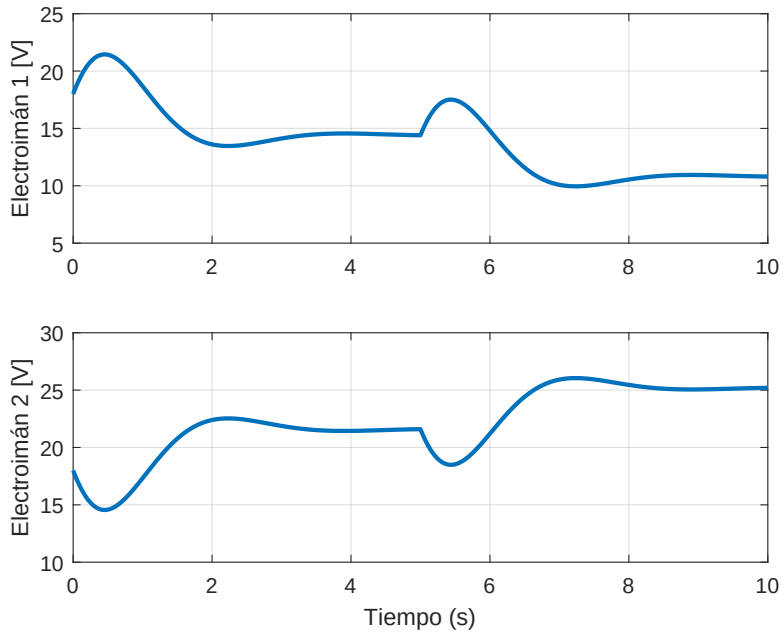

Figura 4.15: Voltajes correspondientes al caso de la figura 4.14.

Tabla 4.1: Variación de la masa m en intervalos de 10g para un setpoint de 10cm.

Masa m (g)	50	60	70	80
Error estacionario (%)	3	4	12	18
Tiempo de asentamiento (s)	2.5	2.5	2.5	2.5
Sobreimpulso (%)	6.1	6.1	6.1	10
Máximo voltaje en electroimanes (V)	28	28	28	30

4.2.5 Desempeño frente a variación de parámetros

Se realizaron experimentos en los que se cambió la masa de prueba movible m. Las experiencias consistieron en ir aumentando progresivamente la masa m en 10g. Los resultados se muestran en la tabla 4.1.

4.2.6 Comparación con otros controladores

El desempeño del controlador propuesto se compara con otros dos controladores cuando se desea alcanzar un setpoint de 10cm. Por un lado, se compara con el controlador que resulta de utilizar el método presentado en [8]. Los enfoques de este trabajo y [8] son iguales con la diferencia que la función de premio utilizada en [8] aplicada al sistema de posicionamiento magnético estudiado es:

$$r(s_t, a_t, s_{t+1}) = \begin{cases} c, & \text{si } \mid x - x_{sp} \mid, \mid y - y_{sp} \mid \leq \epsilon \\ - \mid x - x_{sp} \mid - \mid y - y_{sp} \mid, & \text{en otro caso} \end{cases} \tag{4.1}$$

Donde x e y son las posiciones actuales del sistema de posicionamiento magnético y x_{sp} e y_{sp} son las posiciones finales deseadas. Asimismo, c y ϵ son parámetros de diseño. La curva de aprendizaje obtenida al entrenar el controlador con esta técnica se muestra en la figura 4.2.

Por otro lado, el segundo controlador que se utiliza para la comparación es un controlador no lineal obtenido mediante linealización por realimentación. Sean u_1, u_2, u_3 y u_4 los voltajes que hay que aplicar a cada electroimán de la figura 2.1, las leyes de control vienen dadas por [7]:

$$\begin{aligned} u_1 &= v_0 + \frac{u_1'}{2} \\ u_2 &= v_0 - \frac{u_1'}{2} \\ u_3 &= v_0 + \frac{u_2'}{2} \\ u_4 &= v_0 - \frac{u_2'}{2} \end{aligned} \tag{4.2}$$

Donde u_1' y u_2' se expresan como :

$$\begin{aligned} u_1' &= Rx_3 + 2L\left(\frac{\frac{m}{2K_f}(w_1 + \frac{\beta}{m}\dot{x}_2) - x_2\left(\frac{i_1^2}{(d-x_1)^3} + \frac{i_2^2}{(d+x_1)^3}\right)}{\frac{i_1}{(d-x_1)^2} + \frac{i_2}{(d+x_1)^2}}\right) \\ u_2' &= Rx_6 + 2L\left(\frac{\frac{m}{2K_f}(w_2 + \frac{\beta}{m}\dot{x}_5) - x_5\left(\frac{i_3^2}{(d-x_4)^3} + \frac{i_4^2}{(d+x_4)^3}\right)}{\frac{i_3}{(d-x_4)^2} + \frac{i_4}{(d+x_4)^2}}\right) \end{aligned} \tag{4.3}$$

Asimismo, w_1 y w_2 son las salidas del controlador LQR utilizado para estabilizar el sistema linealizado por realimentación $\dot{\mathbf{z}} = A\mathbf{z} + B\mathbf{w}$, donde las matrices A y B vienen dadas por 2.20. Los detalles de cómo se logra la linealización se encuentran en el capítulo dos de este trabajo. De la misma forma, el significado de cada parámetro de las ecuaciones 4.2 y 4.3 se encuentra en la tabla 2.1.

Tabla 4.2: Comparación entre controladores cuando el setpoint se fija en 10cm.

Parámetro	Control no lineal	[8]	Este trabajo
Error estacionario (%)	1	5	3
Tiempo de asentamiento (s)	2.6	3	2.5
Sobreimpulso (%)	5.2	4.2	6.1
Máximo voltaje en electroimanes (V)	26	25	28

4.3 Discusión

En lo referente al entrenamiento, los resultados mostrados verifican que la propuesta de elegir como función de premio a la ecuación 3.10 junto con la elección de valores finales deseados de la ecuación 3.11 es superior a lo que se propuso en [8] en términos de velocidad de convergencia del algoritmo. Esto se debe a que se le proporciona al controlador información extra del sistema al determinar exactamente cómo debe ser el estado final deseado.

En lo referente al control, los experimentos muestran que el controlador es capaz de seguir setpoints que le fueron mostrados durante el entrenamiento. El controlador neuronal también pudo alcanzar algunos setpoints no vistos durante el aprendizaje. Sin embargo, no se puede determinar qué setpoints no vistos durante el entrenamiento podrán ser controlados. Asimismo, el controlador entrenado logró el seguimiento de una trayectoria cerrada con bastante precisión.

En las pruebas en las que se incluyó un sensor de posición ruidoso, el controlador pudo soportar ruido gaussiano aditivo hasta con una varianza de $\sigma^2 = 0.1$ para tener errores menores al 5%. Asimismo, el controlador pudo soportar una perturbación súbita externa de 5cm en la posición del sistema. Por otro lado, el controlador solo soporta cambios muy leves en la masa movible m, si los cambios son mayores a 20g el error estacionario aumenta por encima del 12%, lo cual se aleja de lo deseado.

En comparación con el controlador no lineal diseñado por linealización por realimentación y el controlador presentado en [8] cuando se desea seguir una posición de 10cm, se observó que el controlador no lineal presenta un mejor desempeño frente a los otros controladores a costa de una dependencia muy fuerte con el modelo del sistema. Los controladores neuronales no alcanzan un desempeño igual al controlador no lineal, pero son suficientemente buenos como para cumplir los requisitos de control con la ventaja que dependen débilmente del modelo del sistema. En el caso de [8] no

se depende del modelo del sistema.

4.4 Limitaciones

El controlador diseñado ha sido capaz de realizar el control del sistema de posicionamiento magnético planteado. Sin embargo, tanto el método de diseño utilizado como el controlador presentan las siguientes limitaciones:

- El algoritmo propuesto para el entrenamiento del controlador requirió del modelo del sistema para poder reducir el número de iteraciones necesarias para alcanzar el comportamiento final.
- El controlador diseñado no contempla ningún mecanismo del tipo adaptivo que le permita mantener un comportamiento adecuado ante cambios significativos en la planta.
- El controlador diseñado no garantiza un desempeño adecuado para todos los setpoints que se le presenten. En general, al controlador no le fue posible controlar todos los setpoints que no vió durante el entrenamiento y no se puede saber de antemano qué setpoints no se podrán controlar. Asimismo, no se tiene una prueba que garantice la estabilidad del sistema.
- En una implementación real es probable que se requiera de un mayor número de iteraciones para el entrenamiento del controlador debido a que planta real puede tener una dinámica más compleja que la reflejada por el modelo utilizado.

Conclusiones

- Se diseñó un neurocontrolador basado en el aprendizaje profundo por refuerzo para el control de un sistema de posicionamiento magnético. El entrenamiento del controlador neuronal se realizó utilizando la arquitectura actor-crítico y el algoritmo DDPG con una función de premio que incluye el estado completo del sistema.

- El enfoque propuesto para el entrenamiento del controlador neuronal logra que el entrenamiento requiera de 100 iteraciones, frente a las 200 iteraciones que requiere el algoritmo DDPG original. Esto reduce a la mitad el tiempo total que requiere el entrenamiento del controlador.

- Se validó el desempeño del controlador propuesto frente al seguimiento de varios setpoints y una trayectoria deseada. El controlador fue capaz de controlar la posición del sistema incluso a setpoints no mostrados durante el entrenamiento.

- Se evaluó la capacidad del controlador de soportar ruido en el sensor de posición, perturbaciones en la posición del sistema y variación de la masa del objeto a mover. El controlador fue capaz de soportar bajos niveles de ruido en el sensor de posición, cambios súbitos significativos en la posición del sistema y leves variaciones en la masa movible.

- Se comparó el desempeño del controlador propuesto frente a un neurocontrolador entrenado con el algoritmo DDPG original y un controlador no lineal basado en la linealización por realimentación. Los tres controladores fueron capaces de cumplir los objetivos de control, siendo el controlador no lineal el de mejor desempeño y el controlador neuronal entrenado con el DDPG original el de menor desempeño. El controlador propuesto mostró un desempeño intermedio entre estos dos. Sin embargo, el controlador no lineal depende fuertemente

del modelo del sistema, mientras que el controlador neuronal entrenado con el algoritmo propuesto tiene una dependencia leve. El neurocontrolador basado en el algoritmo DDPG original no depende del modelo del sistema.

Recomendaciones

Para futuros trabajos se recomienda estudiar lo siguiente:

- Usar redes neuronales del tipo recurrente, similares a las propuestas en [53], para las redes actor y crítico.
- Utilizar la técnica presentada recientemente en [54] que permite utilizar el modelo de la planta dado por 2.9 para mejorar el desempeño del controlador.

Bibliografía

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[2] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," arXiv preprint arXiv:1709.06560, 2017.

[3] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges," *Robotics*, vol. 2, no. 3, pp. 122–148, 2013. [Online]. Available: http://kormushev.com/papers/Kormushev_MDPI_2013.pdf

[4] S. Shead, "30 companies are now making self-driving cars," (Acceso: 01-Julio-2018). [Online]. Available: https://www.businessinsider.com/30-companies-are-now-making-self-driving-cars-2016-4

[5] C. Esparza and R. N. nez, "Controlador adaptativo pd por modelo de referencia para una mesa vibratoria biaxial basada en el mecanismo bielamanivela," (Acceso: 15-Setiembre-2018). [Online]. Available: https://scielo.conicyt.cl/pdf/infotec/v25n2/art21.pdf

[6] E. Bejar and A. Moran, "Deep reinforcement learning based neuro-control for a two-dimensional magnetic positioning system," *Proceedings of the 4th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 268–273, April 2018.

[7] E. Bejar and A. Moran, "Control of a two-dimensional magnetic positioning system with deep reinforcement learning and feedback linearization," *Proceedings of*

the 61st IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), August 2018.

[8] S. Spielberg, R. Gopaluni, and P. Loewen, "Deep reinforcement learning approaches for process control," *Proceedings of the 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, pp. 201–206, May 2017.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations (ICLR)*, 2016.

[10] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," arXiv preprint arXiv:1708.05866, 2017.

[11] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, pp. 550:354–359, 2017.

[12] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," arXiv preprint arXiv:1712.01815, 2017.

[13] L. Kaelbling and M. Littman, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[14] M. Minar and J. Naher, "Recent advances in deep learning: An overview," arXiv preprint arXiv:1807.08169 , 2018.

[15] D. O. Hebb, "The organization of behavior; a neuropsychological theory," 1949.

[16] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, 65.6(1958): 386.

[17] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," 1974.

[18] R. Vidal, J. Bruna, R. Giryes, and S. Soatto, "Mathematics of deep learning," arXiv preprint arXiv:1712.04741, 2017.

[19] S. Amarjyoti, "Deep reinforcement learning for robotic manipulation-the state of the art," arXiv preprint arXiv:1701.08878, 2017.

[20] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems 25*, pp. 1106–1111, 2012.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[22] I. Arel, "Deep reinforcement learning as foundation for artificial general intelligence," in *Theoretical Foundations of Artificial General Intelligence*, P. Wang and B. Goertzel, Eds. Springer, 2012, ch. 6, pp. 89–102.

[23] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, May-June 2017.

[24] K. J. Astrom and T. Hagglund, "The future of pid control," *Control Engineering Practice*, vol. 9, pp. 1163–1175, November 2001.

[25] M. T. Munir, I. A. Udugama, I. A. Boiarkina, W. Yu, and B. R. Young, "Beyond the theory - how can academia contribute to the advanced process control of industrial processes?" *Proceedings of the 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, pp. 541–546, May 2017.

[26] A. Farahmand, S. Nabi, and D. N. Nikovski, "Deep reinforcement learning for partial differential equation control," *Proceedings of the 2017 American Control Conference*, pp. 3120–3127, May 2017.

[27] Y. Li and Q. Xu, "Design and analysis of a totally decoupled flexure-based xy parallel micromanipulator," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 645–657, June 2009.

[28] S. Xiao, Y. Li, and J. Liu, "A model reference adaptive pid control for electromagnetic actuated micro-positioning state," *Proceedings of the 8th IEEE International Conference on Automation Science and Engineering*, August 2012.

[29] G. Spencer and M. Hagmann, "Development of a scanning tunneling microscope for the carrier profiling of semiconductors by scanning frequency comb microscopy," *IEEE Workshop on Microelectronics and Electron Devices (WMED)*, April 2017.

[30] S. Xiao, Y. Li, and X. Zhao, "Design and analysis of a novel flexure-based xy micro-positioning stage driven by electromagnetic actuators," *Proceedings of 2011 International Conference on Fluid Power and Mechatronics*, pp. 953–958, August 2011.

[31] R. Zhang, X. CHen, J. Lu, S. Wen, S. Nepal, and Y. Xiang, "Using ai to hack ia: A new stealthy spyware against voice assistance functions in smart phones," arXiv preprint arXiv:1805.06187 , 2018.

[32] G. Tesauro, "Temporal difference learning and td-gammon," *Communications of the ACM*, 38(3):58-68, 1995.

[33] J. C. Hoskins and D. M. Himmelbau, "Process control via aritificial neural networks and reinforcement learning," *Computer and Chemical Engineering*, vol. 16, pp. 241–251, April 1992.

[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*, 2nd ed. MIT Press, 1998.

[35] R. Sadiq and M. Khan, "Analyzing self-driving cars on twitter," arXiv preprint arXiv:1804.04158 , 2018.

[36] E. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed. Springer, 2007.

[37] S. Verma, W. jong Kim, and H. Shakir, "Multi-axis maglev nanopositioner for precision manufacturing and manipulation applications," *IEEE Transactions on Industrial Applications*, vol. 41, no. 5, pp. 1159–1167, September/October 2005.

[38] M.-Y. Chen, H.-H. Huang, and S.-K. Hung, "A new design of a submicropositioner utilizing electromagnetic actuators and flexure mechanism," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 1, pp. 96–106, January 2010.

[39] A. Forrai, T. Ueda, and T. Yumura, "Electromagnetic actuator control: A linear parameter-varying (lpv) approach," *IEEE Transactions on Industrial Applications*, vol. 54, no. 3, pp. 1430–1441, June 2007.

[40] B. Razavi, *Design of Analog CMOS Integrated Circuits*, 1st ed. McGraw-Hill Education, 2000.

[41] R. Hasan, "Influence of device performance of sub-10 nm gan-based dg-mosfets over conventional si-based sg-mosfets," *4th International Conference on Advanced in Electrical Engineering (ICAEE)*, September 2017.

[42] Y. Tsividis and C. McAndrew, *Operation and Modeling of the MOS Transistor*, 3rd ed. Oxford University Press, 2010.

[43] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, vol. 2, March 1991.

[44] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, pp. 4–27, March 1990.

[45] F. Lin and P. Chou, "Adaptive control of two-axis motion control system using interval type-2 fuzzy neural network," *IEEE Transactions on Industrial Electronics*, vol. 56, pp. 178–193, January 2009.

[46] P. M. Larsen, "Industrial applications of fuzzy logic control," *International Journal of Man-Machine Studies*, vol. 12, pp. 3–10, January 1980.

[47] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2001.

[48] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *International Conference on Learning Representations (ICLR)*, 2014.

[49] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.

[50] M. Hausknecht and P. Stone, "Deep reinforcement learning in parametrized action space," *International Conference on Learning Representations (ICLR)*, 2016.

[51] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical Review*, 36(5):823, 1930.

[52] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," *International Conference of Machine Learning (ICML)*, 2010.

[53] Y. Wang, K. Velswamy, and B. Huang, "A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems," *Processes*, vol. 5, no. 3, 2017.

[54] V. Pong, S. Gu, M. Dalal, and S. Levin, "Temporal difference models: Model-free deep rl for model-based control," *International Conference on Learning Representations (ICLR)*, 2018.