Universidad Michoacana de San Nicolás de Hidalgo
Facultad de Ingeniería Eléctrica
División de Estudios de Posgrado

Control de aprendizaje por refuerzo con el algoritmo actor crítico de ventaja

TESIS

Para obtener el grado de
Maestro en Ciencias en Ingeniería Eléctrica

Presenta
Miguel Angel Pimentel Vallejo

Dr. Roberto Tapia Sánchez
Director de Tesis

Agosto 2022

La dedico a todos aquellos que fallecieron durante la realización de este trabajo en especial a mi tita, que con su cariño, su forma de ser y apoyo tanto moral como económico tuvo tanto que ver en el desarrollo de mi vida.

Agradezco a todos los que me apoyaron durante la realización de esta tesis, mi mamá y hermana por darme aliento, a mi novia Itzel por creer en mí y apoyarme, a mi asesor el Dr. Roberto Tapia por brindarme todos los medios para la realización de la tesis y por sus críticas.

Resumen

En esta tesis se presentan los resultados obtenidos de implementar un controlador basado en aprendizaje por refuerzo, en particular se utiliza el algoritmo de actor crítico de ventaja.

El método es entrenado para controlar dos sistemas:

- Motor de corriente directa. El cual solamente se controla de manera simulada utilizando su modelo lineal.
- Péndulo invertido rotatorio. Este sistema se controla de manera simulada usando su modelo no lineal y de manera real, usando un módulo para experimentación de la marca Quanser.

Una vez entrenados, los controladores son sometidos a una serie de pruebas:

- Seguimiento de una referencia que no cambia con el tiempo.
- Seguimiento de una señal de referencia con cambios de valor a través del tiempo.
- Seguimiento de una señal senoidal.

Con estas pruebas se compara el desempeño del controlador entrenado con distintas funciones de recompensa, entre las cuales se utiliza la propuesta en esta tesis, esto con la finalidad de mostrar cuál tiene el mejor desempeño.

Por último se presentan las conclusiones y los trabajos futuros.

Palabras clave: rotatorio, péndulo, invertido, motor, corriente directa

Abstract

In this thesis the results obtained from implementing a controller based on reinforcement learning are presented, in particular the algorithm of critical actor of advantage is used.

The method is trained to control two systems:

- Direct current motor. Which is only controlled in a simulated way using its linear model.
- Rotating inverted pendulum. This system is controlled in a simulated way using its non-linear model and in a real way, using a module for experimentation of the Quanser brand.

Once trained, controllers are subjected to a series of tests:

- Tracking a reference that does not change over time.
- Tracking a reference signal with value changes over time.
- Tracking a sinusoidal signal.

With these tests, the performance of the trained controller is compared with different reward functions, among which the proposal in this thesis is used, in order to show which one has the best performance.

Finally, conclusions and future work are presented.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

$\mathbb{R}$	Números reales.
$\mathbb{N}$	Números naturales.
$\in$	Pertenece a.
$\subset$	Es un subconjunto de.
k	Número natural que simboliza la iteración en la que va el entrenamiento.
s	Estado.
s_k	Estado en la k-ésima iteración.
a_k	Acción en la k-ésima iteración.
r_k	Recompensa en al k-ésima iteración.
u_k	Señal de control en al k-ésima iteración.
s_{k+1}	Estado en la siguiente iteración.
a_{k+1}	Acción en la siguiente iteración.
r_{k+1}	Recompensa en la siguiente iteración.
u_{k+1}	Señal de control en la siguiente iteración.
Δu	Cambio en la señal de control.
r	Recompensa.
$\mathcal{R}$	Conjunto de todas las recompensas posibles.
$\mathbb{S}$	Conjunto de todos los estados posibles.
$\mathbb{A}$	Conjunto de todas las acciones posibles.
$R(*)$	Función de recompensa.
$\pi(s)$	Política del agente.
V^{π}	Función evaluadora del estado.
$A(s_{k+1}, s, r)$	Función de ventaja.
lr	Taza de aprendizaje.
$Loss_{actor}$	Función de pérdida de la red neuronal actor.
$Loss_{critico}$	Función de pérdida de la red neuronal crítico.
x	Variable de estado de un sistema dinámico.
$\dot{x}$	Derivada de la variable de estado.
x_n	Enésima variable de estado.
$\dot{x}_n$	Enésima derivada de la variable de estado.
x_{ref}	Referencia de la variable de estado.
x_{ref_n}	Enésima referencia de la variable de estado.
X	Vector formado por las variables de estado.
X_{ref}	Vector formado por las referencias de las variables de estado.

ω	Velocidad del motor de corriente continua.
i	Corriente en la armadura del motor.
θ	Posición angular del brazo.
$\dot{\theta}$	Velocidad angular del brazo.
α	Posición del péndulo.
$\dot{\alpha}$	Velocidad angular del péndulo.
c_n	Enésimo parámetro constante para la función de recompensa.
C	El valor máximo de la función de recompensa.
ϵ	Parámetro constante para la función de recompensa para la tolerancia.
H	Entropía de la toma de decisiones.
k_H	Ganancia de la entropía.
N	Cantidad de variables de estado a controlar.
e_{n_k}	Enésimo error antes de aplicar la acción.
$e_{n_{k+1}}$	Enésimo error después de aplicar la acción.
γ	Constante que cuantifica la importancia de las acciones pasada.
Δt	Periodo de tiempo en que se lleva a cabo el algoritmo.
t_f	Duración máxima de cada episodio.
$r_{epi_{max}}$	Recompensa máxima por episodio.
r_{max}	Recompensa máxima de la función de recompensa.
$a_{totales}$	Número de acciones tomadas durante un episodio.
R_m	Resistencia de armadura del péndulo.
K_m	Constante de FEM del péndulo.
K_t	Constante de torque del motor del péndulo.
n_g	Eficiencia de los engranajes.
n_m	Eficiencia del motor del péndulo.
L_p	Longitud del péndulo.
L_r	Inductancia de armadura del motor del péndulo.
m_p	Masa del péndulo.
J_p	Inercia del péndulo.
J_r	Inercia del brazo.
B_p	Viscosidad péndulo.
B_r	Viscosidad brazo.
g	Constante de la gravedad.
L_a	Inductancia de armadura.
R_a	Resistencia de armadura.
K_{gm}	Constante de FEM.
K_a	Constante de torque.
f	Constante de fricción.
J	Momento de inercia del motor.

Lista de Abreviaturas

A2C Advantage Actor Critic.

A3C Asynchronous Advantage Actor Critic.

CD Corriente Directa.

FEM Fuerza electromotiz.

MDP Markov decision processes.

PID Proportional Integral Derivative control.

RL Reinforcement learning.

TD Temporal difference learning.

Capítulo 1

Introducción

1.1. Planteamiento del Problema

Los métodos de aprendizaje reforzado o en inglés reinforcement learning (RL), son algoritmos que están inspirados en la forma en que los humanos aprenden. Estos métodos por aprendizaje reforzado son utilizados para realizar actividades complejas como las que solamente una persona podría hacer. Teniendo aplicaciones en conducción autónoma, robótica, sector salud, ecología, artes gráficas, en redes inteligentes, entre muchas más. En cada una de estas áreas las actividades que desarrolla el algoritmo son diferentes, van desde tomar decisiones, esto en el área de sector salud, hasta realizar una animación entera, en el área de artes gráficas. [François-Lavet et al., 2018]

Un ejemplo más específico podría ser AlphaGo que usa el aprendizaje reforzado para aprender a jugar Go, este es juego de mesa asiático, el cual tiene la particularidad de que ofrece una gran cantidad de posibles movimientos para los jugadores. AlphaGo ha logrado vencer con éxito incluso a jugadores expertos demostrado así la capacidad que tienen estos métodos de aprendizaje reforzado para realizar actividades mejor que los seres humanos. Otro ejemplo podría ser AlphaStar que al igual que AlphaGo es basado en aprendizaje reforzado, en este caso en un ambiente de un videojuego de estrategia basado en un mundo libre donde las posibles acciones podrían ser infinitas, llamado StarCraft II. AlphaStar

también ha sido probada contra seres humanos, logrando vencer a jugadores profesionales. Tanto AlphaGo como AlphaStar han sido desarrollados por el grupo de investigadores de Google, llamado Deepmind, los cuales son los que actualmente tienen los resultados más llamativos en este campo. [Silver et al., 2018] [Arulkumaran et al., 2019]

Si bien se ha mencionado antes que el aprendizaje reforzado ha tenido una variedad de aplicaciones, el interés de la presente investigación se centra en el uso de estos algoritmos para resolver problemas de control. También es importante mencionar que hay varios métodos con que emplean aprendizaje reforzado, la presente tesis se enfoca en el llamado actor crítico, el cual como su nombre lo dice trata de imitar la forma en que interactúan un actor, que es el que toma las decisiones y el crítico, que evalúa estas, más adelante se describe de manera más adecuada su funcionamiento.

Determinado el método a usar y la aplicación, se puede realizar una revisión del estado del arte, así el primero que se cita [Si and Wang, 2001], este consigue controlar un sistema de péndulo invertido de uno, dos y tres brazos usando el método de actor crítico, para este caso el actor es una red neuronal y el crítico es una función propuesta, los resultados obtenidos son el control de estos sistemas. Otra investigación más reciente es el de [Spielberg et al., 2017] donde utilizan un método parecido al anterior, agregando una memoria donde se van almacenando datos de acciones anteriores, esto se utiliza para mejorar el desempeño del algoritmo, en este caso se implementó el control para sistemas lineales. En [Grondman et al., 2011] utilizan una mezcla entre el método actor crítico y regresión lineal. Por último en [Bejar and Moran, 2018] usan una mezcla entre los métodos de actor crítico utilizados en [Spielberg et al., 2017] y [Sutton and Barto, 2018], donde usan también una memoria, pero proponen una función para el crítico diferente, obteniendo buenos resultados, para las referencias vistas durante el entrenamiento, el sistema utilizado para la realización de esta investigación es uno de posicionamiento magnético.

Como se menciona en el párrafo anterior hay diversas maneras de implementar el algoritmo de actor crítico, para esta tesis se usa la implementación llamada actor crítico

de ventaja, que por su nombre en inglés se le conoce como Advantage Actor Critic (A2C), este a diferencia de los anteriores, tanto el actor como el crítico son redes neuronales y su nombre proviene de que buscan obtener una ventaja de cada acción tomada. Actualmente, en la plataforma de artículos de carácter académico Google Scholar, pueden encontrarse una gran cantidad de documentos recientes sobre este tema, sin embargo la mayoría son enfocados en el control del péndulo invertido, un ejemplo es [Zheng et al., 2020], donde controlan el sistema ya antes mencionado, obteniendo buenos resultados, aunque dejando interrogantes acerca de como seleccionar ciertas funciones o parámetros para que funcione el algoritmo. El principal problema que se detecta es la poca experimentación en diferentes tipos de sistemas, lo que genera que haya poca información sobre el uso de este algoritmo para los demás sistemas de control.

1.2. Marco Teórico

Las bases del entrenamiento reforzado se encuentran en la toma de decisiones, este fenómeno ha despertado en una gran cantidad de interés en diferentes disciplinas como la filosofía, psicología, economía, política y ciencia de computadoras. La teoría de la decisión estudia las condiciones, motivaciones, fenómenos psicológicos, y comportamientos de aquellos que toman decisiones. Debido a que esta teoría es de interés de múltiples disciplinas se tiene aportaciones de diversos autores, los cuales coinciden en que la teoría de la decisión tiene dos enfoques: el normativo y el descriptivo. El enfoque normativo se basa en que hay un tomador de decisiones ideal, el cual se encuentra en un entorno del cual puede obtener la información necesaria para tomar una decisión totalmente racional. El enfoque descriptivo trata de describir las decisiones del tomador de decisiones asumiendo que este se gobierna por unas reglas bien definidas [Peterson, 2017][MacCrimmon, 1968].

Dentro de la teoría de la decisión hay términos que tienen un significado muy preciso como: decisión racional, buena decisión, decisión de riesgo y decisión bajo ignorancia. Se le llama a una decisión racional a la que hace el tomador de decisiones que es la más comprensible de tomar. Una decisión buena es la que mejor resultados da en comparación

con las otras decisiones. Una decisión puede ser buena sin ser racional o viceversa. Decisión de riesgo significa que el hacedor de decisiones conoce que hay posibles resultados negativos. Decisión bajo ignorancia donde las posibilidades de resultados negativos son desconocidas [Peterson, 2017].

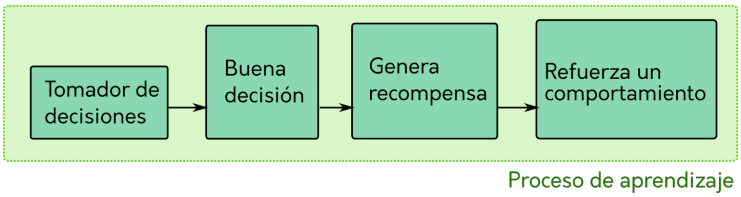

Figura 1.1: Diagrama a bloques del proceso de aprendizaje según la teoría del aprendizaje por refuerzo

Una vez definido lo que enmarca la toma de decisiones se procede con la teoría del aprendizaje. Se define el aprendizaje como el proceso mediante el cual se adquieren habilidades, conocimientos, valores, actitudes y reacciones emocionales. Los resultados de diversas investigaciones sentaron las bases y los principios de la teoría del aprendizaje. Uno de los principios de esta teoría es que el aprendizaje se llevaba a cabo a través de una serie de decisiones, las cuales si son buenas decisiones generan una recompensa, esto hace que un tipo de comportamiento sea reforzado con el tiempo. Este principio de que la recompensa sigue al comportamiento, nació de observar muchas situaciones en la naturaleza como por ejemplo, a los animales que logran entrenar a base de darles una recompensa por realizar un comportamiento deseado, y finalmente logran repetir el comportamiento sin que se le dé la recompensa [Ormrod, 2011]. En la Figura 1.1 se muestra un diagrama de bloques del proceso de aprendizaje antes mencionado.

Los modelos matemáticos que intentan imitar el proceso de decisión y aprendizaje, son las bases del RL. La primera herramienta matemática con la que se inició fue el proceso de decisión de Markov o por sus siglas en inglés Markov decision processes (MDP), que como su nombre lo indica intentan imitar el proceso de decisión. Los MDP son mapas donde de manera matemática se expresan todas las posibles decisiones de una manera secuencial

[Sutton and Barto, 2018]. En la Figura 1.2 se puede ver un ejemplo de lo que sería un MDP.

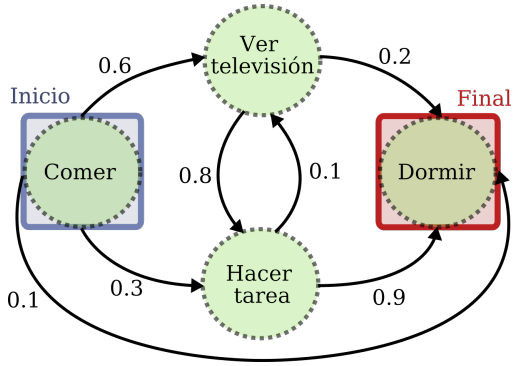

Figura 1.2: Ejemplo de MVP sobre qué hacer en una tarde

Como se puede ver en la Figura 1.2 en forma de círculos se determina cada una de las posiciones que se pueden tomar, a las cuales se conocen como estados. Los posibles cambios de estado están denotados de manera probabilista, donde el máximo valor es uno y mínimo cero. La probabilidad del cambio de estado solamente depende de la posición actual, más no de las pasadas. Este método funciona cuando se conoce totalmente el entorno donde se debe de tomar la decisión [Sutton and Barto, 2018].

Se le puede dar un enfoque agente-ambiente al MDP para intentar imitar el proceso de aprendizaje. Se le llamará al tomador de decisiones como agente y a todo lo que interactúa con él se le llamará ambiente. El agente interactúa con el ambiente por un número de iteraciones k hasta que se haya logrado entrenar al agente, donde $k \in \mathbb{N}$. La forma en la que interactúa el agente se le llama acción. Se define a_k como la acción en la k-ésima iteración, al conjunto de todas las acciones se le conoce como $\mathbb{A}$. Se premiarán las acciones del agente que sean las deseadas por medio de una recompensa. Esta se define r_k como la recompensa en la k-ésima iteración, al conjunto de todas las recompensas se le conoce como $\mathcal{R}$. Los estados en los que se encuentra el agente en la iteración k se definen como s_k, al conjunto de todos los estados posibles se le denominan como $\mathbb{S}$. Se muestra en la Figura 1.3 un esquema de bloques de cómo interactúan el ambiente y el agente [Sutton and Barto, 2018].

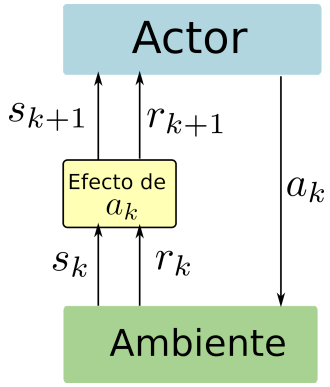

Figura 1.3: Diagrama de interacción entre el agente y el ambiente

En el diagrama de la Figura 1.3 aparecen los términos recompensa futura r_{k+1} y estado futuro s_{k+1}, donde por futura se entiende que es después de aplicar la acción a_k. [Sutton and Barto, 2018].

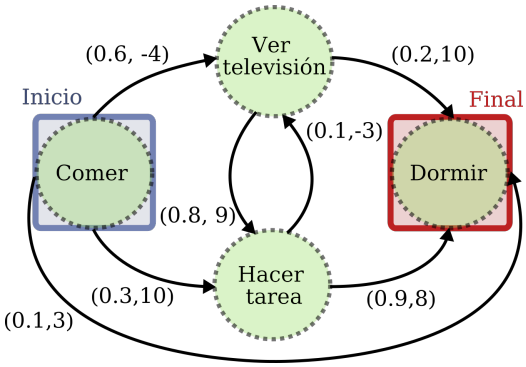

Figura 1.4: Ejemplo de MVP con enfoque agente-ambiente sobre qué hacer en una tarde

Si al MVP de la Figura 1.2 se le da el enfoque agente-ambiente como resultado tendrá lo mostrado en la Figura 1.4, donde entre cada transición aparece la probabilidad del cambio de estado junto con su recompensa. Entre mayor sea la recompensa más se va a reforzar este comportamiento. Las recompensas están ligadas a las metas que se desean conseguir, para el ejemplo lo que se desea conseguir es que la persona priorice hacer su tarea antes que ver televisión [Sutton and Barto, 2018].

1.3. Estado del Arte

El algoritmo A2C ha sido implementando para jugar videojuegos, esto debido a que ofrecen un entorno difícil de comprender y pone a prueba el algoritmo ante una situación parecida a la realidad [Wu and Tian, 2016]. También es utilizado para administrar el tráfico de las ciudades [Xiong et al., 2019].

Otro uso que se le da al algoritmo A2C, es el control de sistemas, en la mayoría de los casos se implementa para controlar un péndulo invertido [Zheng et al., 2020], se puede encontrar que se mezcla el algoritmo A2C con técnicas ya probadas en control, como por ejemplo con el control proporcional, integrador y derivador PID, donde el algoritmo sintoniza al controlador, es decir escoge los parámetros, para cada situación, esto generalmente se hace por medio de métodos numéricos basados en pruebas de la planta, también estos parámetros suelen ser constantes, pero con el algoritmo A2C no es necesario hacer pruebas, ya que el algoritmo lo hace por sí mismo y otra particularidad es que los parámetros los va cambiando el algoritmo con el tiempo, con estos cambios el control PID tiene un mejor desempeño e incluso es capaz de seguir con precisión referencias que varían con el tiempo, cosa que normalmente no puede hacer [Sun et al., 2019a] [Sun et al., 2019b].

Otros trabajos que se deben mencionar sobre el estado del arte del algoritmo A2C en sistemas de control están mencionados en el planteamiento del problema. Cabe mencionar que actualmente hay un algoritmo que se podría decir que es una evolución de Advantage Actor Critic (A2C), llamado actor crítico de ventaja asíncrono que es más conocido por su nombre en inglés Asynchronous Advantage Actor Critic (A3C), la mayor diferencia es que en el algoritmo A3C interactúan múltiples agentes cada uno con su propio ambiente, lo cual hace que el entrenamiento sea rápido [Sun et al., 2019b] [Sun et al., 2019a].

1.4. Objetivos de la Tesis

1.4.1. Objetivo general

- Realizar el control de un motor de corriente directa y un péndulo rotacional invertido, por medio del algoritmo A2C

1.4.2. Objetivos particulares

- Realizar el entrenamiento de dos redes neuronales utilizando el algoritmo de A2C, con la finalidad de controlar un motor de corriente directa y un péndulo rotacional invertido.
- Implementar el controlador entrenado con A2C.
- Validar el controlador basado en A2C, con múltiples puntos y trayectorias de referencia.
- Probar el desempeño del controlador basado en A2C, frente ruido y perturbaciones externas.

1.5. Justificación

En el estado del arte de esta tesis, se encuentra que todas las investigaciones con el método A2C están enfocadas el péndulo invertido y que solamente han sido implementadas en simulación, los puntos antes mencionados hacen que no haya suficiente información para saber si el controlador basado en A2C es viable para diferentes tipos de sistemas o su implementación experimental tenga validez, esto justifica la realización de esta tesis, ya que aportará a los principales problemas detectados en el estado del arte.

En el estado del arte de esta tesis, se encuentran los siguientes puntos donde este trabajo puede hacer una aportación:

- El algoritmo A2C ha sido utilizado solamente en el péndulo invertido, en esta tesis se utiliza en dos sistemas diferentes, un motor de corriente directa y un péndulo

rotacional invertido, con el objetivo de verificar la viabilidad de aplicar el algoritmo a otros sistemas.

- Los controladores basados en A2C no han sido implementados en forma física, en esta tesis se implementa. Lo anterior permite conocer cuáles son los retos en implementar este tipo de algoritmos.

- Hay pocas funciones de recompensa orientadas que se utilizan en controladores entrenados por refuerzo, por lo que se propone una y se compara con las del estado del arte.

1.6. Hipótesis

Es posible realizar un control basado en A2C. Este control toma decisiones adecuadas para mantener el sistema en la referencia a pesar de que se presenten ruido en los sensores, perturbaciones debido a situaciones externas o que la referencia este cambiando con el tiempo. Para la realización y entrenamiento del controlador basado en A2C no es necesario conocer el modelo del sistema.

1.7. Variables de Estudio

Las variables de estudio son las señales de las magnitudes físicas del sistema a controlar así como la mejora del entrenamiento del controlador. Las señales provenientes del sistema son evaluadas constantemente junto con la referencia del control, esto para medir la efectividad del controlador. Entre más se parezca la magnitud física del sistema a la referencia mejor es evaluado el controlador. El sistema puede tener más variables físicas de las que se van a controlar, pero estas no servirán para evaluar el desempeño del controlador basado en A2C.

1.8. Preguntas de Investigación

Las preguntas que se plantean al realizar esta tesis son:

¿Cómo es el desempeño de control basado en A2C?.

¿Se podrá entrenar un el controlador basado en A2C sin conocer el modelo del sistema?.

¿El controlador podrá rechazar las perturbaciones externas?.

¿Podrá el controlador seguir referencias variantes con el tiempo?.

¿Es en realidad una técnica de control aplicable para cualquier sistema?.

¿Cómo se escoge la función de recompensa de un sistema?.

1.9. Metodología

La metodología a utilizar en esta tesis es del tipo cuantitativa debido a que se puede comprobar la hipótesis planteada por medio de experimentos donde se miden las variables físicas.

Una vez desarrollado el controlador basado en A2C se valida su funcionamiento sometiéndose a diferentes pruebas. Durante las pruebas se miden parámetros físicos de la planta por medio de sensores, en caso de no poderse realizar físicamente el experimento se comprobará por medio de simulaciones con el modelo de la planta.

Se almacena la información de cada prueba realizada, para presentarla como evidencia. Una vez con la información procesada, se analizará el desempeño del controlador por medio del cálculo del error a lo largo de la prueba. El análisis dará como resultado si se comprueba o no las hipótesis propuestas.

1.10. Descripción de Capítulos

A continuación se describen los capítulos de la tesis:

Capítulo 1: Se presenta el tema a investigar, las motivaciones de seleccionar el tema, el marco en el que se desarrolla y se plantea un esquema de cómo es que se quiere llevar a cabo la investigación

Capítulo 2: En este capítulo se describe los detalles de como es que se llevaron a cabo la pruebas, se explica el método utilizado así como todas las consideraciones que se necesitaron para utilizarlo, que tipos de funciones de recompensa se utilizan, las condiciones de las diferentes pruebas y los modelos utilizados.

Capítulo 3: En este capítulo se presentan los resultados de la evolución, así cómo pruebas realizadas a los controladores para observar su comportamiento en diferentes condiciones.

Capítulo 4: En este capítulo se presentan las conclusiones personales obtenidas a través de la observación de los experimentos realizados durante este trabajo, también se habla de cuáles son los posibles trabajos a realizar a futuro a partir de lo que se descubrió en este trabajo.

Capítulo 2

Proceso de aprendizaje

2.1. Introducción

Este capítulo se presenta el proceso de aprendizaje del controlador, se expone el funcionamiento del método A2C, la implementación de este, y ademas se presentan las funciones de recompensa con las cuales se hace la comparación. Adicionalmente se mencionan los parámetros necesarios que se requieren definir, cuales son los valores utilizados para cada experimento o prueba.

2.2. Método Actor Crítico de Ventaja

Todo lo descrito en esta sección está basado en [Sutton and Barto, 2018] y [Laura Graesser, 2019].

El algoritmo actor crítico de ventaja A2C, consta de dos redes neuronales que interactúan entre sí, para realizar el proceso de aprendizaje, una de las redes neuronales es la encargada de tomar las acciones a_k, a esta red neuronal se le denomina actor, mientras la otra red neuronal a la cual se le conoce como crítico, califica las acciones tomadas por el actor. Parte del aprendizaje del actor y del crítico se basa en una función de recompensa que se debe de proponer según el comportamiento que se desee reforzar. De manera general este método, como la mayoría basados en aprendizaje reforzado, mide su desempeño de

acuerdo a la máxima recompensa obtenida durante el entrenamiento. Se puede decir que el fin último del método A2C siempre es maximizar la función de recompensa.

La principal característica de este método de aprendizaje reforzado es que es tanto "basado en valor" como en "política". Los métodos de aprendizaje reforzados basados en valor son aquellos que proponen una función evaluadora del estado V^{π}, la cual se busca a través de la exploración. La función evaluadora de estados califica la mejor acción para el estado dado y con esto saber la mejor acción a tomar. Los métodos basados en política como su nombre lo dice se propone una función llamada política $\pi(s)$ la cual da a cada estado una acción.

El crítico es entrenado por medio de la función de ventaja $A(s_{k+1}, s, r)$, esta función busca las mejores críticas a las acciones basándose en las acciones pasadas, entiéndase que por "mejores críticas" se refiere a las que benefician alcanzar mayores recompensas. La función de ventaja se calcula por medio del método de aprendizaje por diferencia temporal mejor conocido por su nombre en inglés como Temporal difference learning (TD). La función se muestra en la ecuación (2.1).

$$A(s_{k+1}, s_k, r) = r + \gamma V^{\pi}(s_{k+1}) - V^{\pi}(s_k) \tag{2.1}$$

En la ecuación (2.1), r es la recompensa, $V^{\pi}(s_k)$ es la función evaluadora de estado antes de tomar la acción, $V^{\pi}(s_{k+1})$ es la función evaluadora de estado después de tomar la acción y γ es el factor de descuento, el cual indica cuanto se toma en cuenta las acciones pasadas. Para el caso del algoritmo A2C la función $V^{\pi}(s_k)$ será el valor otorgado por la red neuronal conocida como crítico. Con la función de ventaja A realiza el cálculo para la retroalimentación de la red neuronal tanto del crítico como del actor. La recompensa r es otorgada por la función de recompensa propuesta $R(*)$.

La red neuronal del crítico tiene varias entradas, estas corresponden a información que se considera necesaria para realizar el control o cualquier función que se desee, estas entradas pueden ser asignadas a criterio de quien diseña la red neuronal. Por ejemplo para el caso de control podrían ser las señales de las variables de estados. Al contrario de las entradas de la red neuronal como salida solamente tendrá una, que es evaluación de la acción tomada. Se ejemplifica con un diagrama la red neuronal del crítico en la Figura 2.1

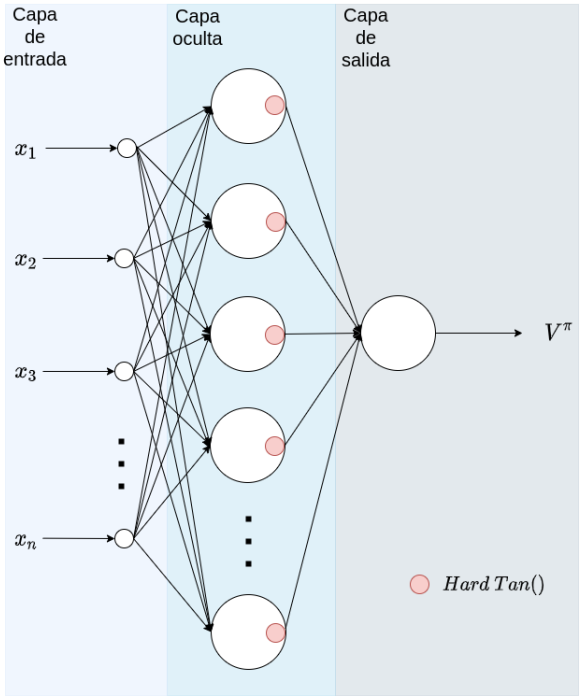

Figura 2.1: Esquema general de la red neuronal usada como crítico

La función que otorga la política a seguir $\pi(s)$, es decir la función que aplica la acción de control, en este método es dada el actor, por lo cual asigna una acciones a cada estado en el que se encuentre el sistema, en este caso las señales provenientes de los sensores de la planta. La red neuronal del actor al igual que la del crítico tiene varias entradas, la diferencia es que la red neuronal del actor tiene múltiples salidas las cuales representan cada acción posible.

Se debe mencionar que este método de actor crítico tiene dos versiones, uno donde toma acciones discretas y el otro las acciones continuas. Para la versión que toma acciones discretas, las acciones ya están bien definidas, es decir que el conjunto de acciones $\mathbb{A}$, está compuesto por una cantidad definida de acciones $[a_1, a_2, ...a_n]$, cada una de estas es representada con una salida en la red neuronal del actor, el valor numérico que arroje la salida representa la probabilidad de que pase esta acción, la que tenga la mayor probabilidad es la acción escogida, por lo tanto del conjunto de acciones solamente una será tomada. Por otro lado, cuando las acciones son continuas el conjunto de acciones $\mathbb{A}$ se podría ver como infinito por lo que las acciones se toman directamente como el valor numérico a la salida de la red neuronal y al igual que para la versión discreta puede haber las salidas que sean necesarias. Se puede ver representada esta red neuronal del actor de manera general en la Figura 2.2.

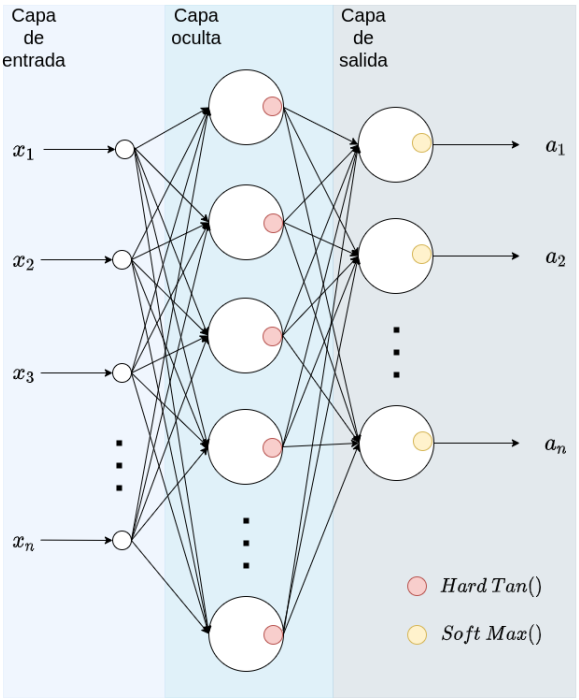

Figura 2.2: Esquema general de la red neuronal usada como actor

Como redes neuronales, el actor y crítico tienen ciertas características para su re-

troalimentación tal como: la función de pérdida, la cual sirve para calcular el error que se desea optimizar. La función de pérdida para el crítico es la mostrada en la ecuación (2.2), esta depende totalmente de la función de ventaja $A(s_{k+1}, s_k, r)$.

$$Loss_{critico} = \frac{1}{2}(A(s_{k+1}, s_k, r))^2 + Hk_H \tag{2.2}$$

La función de pérdida de la red neuronal del actor se muestra en la ecuación (2.3), la cual depende de la función de ventaja $A(s_{k+1}, s_k, r)$ y de la acción seleccionada.

$$Loss_{actor} = -log(\pi(s))A(s_{k+1}, s_k, r) + Hk_H \tag{2.3}$$

A estas funciones de pérdida mostradas en la ecuaciones (2.2, 2.3), se les agrega la entropía de la probabilidad de las acciones "H" multiplicado por una ganancia "k_H" tal como se realiza en [Wu and Tian, 2016], esto ayuda a que el crítico y actor no lleguen a sobre optimizarse en partes diferentes, lo que podría ocasionar un aprendizaje deficiente.

Otro aspecto a tener en cuenta es el método que se utilizará para la optimización del error de las redes neuronales. En esta tesis se utiliza el optimizador de Adam, el cual solamente tiene un parámetro configurable llamado tasa de aprendizaje lr, este pondera el cambio que tendrá los parámetros de la red neuronal en proporción al valor otorgado por función de pérdida.

El método comienza proporcionando valores aleatorios a las redes neuronales, después se define la cantidad de episodios o épocas, es decir las veces que el controlador y el sistema interactúan por un periodo de tiempo para aprender, se deberá cumplir con cada uno de ellos para terminar el entrenamiento. El procedimiento a continuación descrito es

repetido de manera constante hasta finalizar el episodio, se toma una acción a de la red neuronal del actor, esta se aplica al sistema para obtener el nuevo estado, se obtiene la recompensa r de la función de recompensa $R(*)$, se calcula la función de ventaja para después con esta obtener los valores de las funciones de pérdida tanto del actor como del crítico y después se actualizan las redes neuronales, finalmente se asigna el nuevo estado al estado actual, esto se repite hasta terminar el episodio. Este procedimiento queda mejor plasmado en diagrama de flujo mostrado en la Figura 2.3.

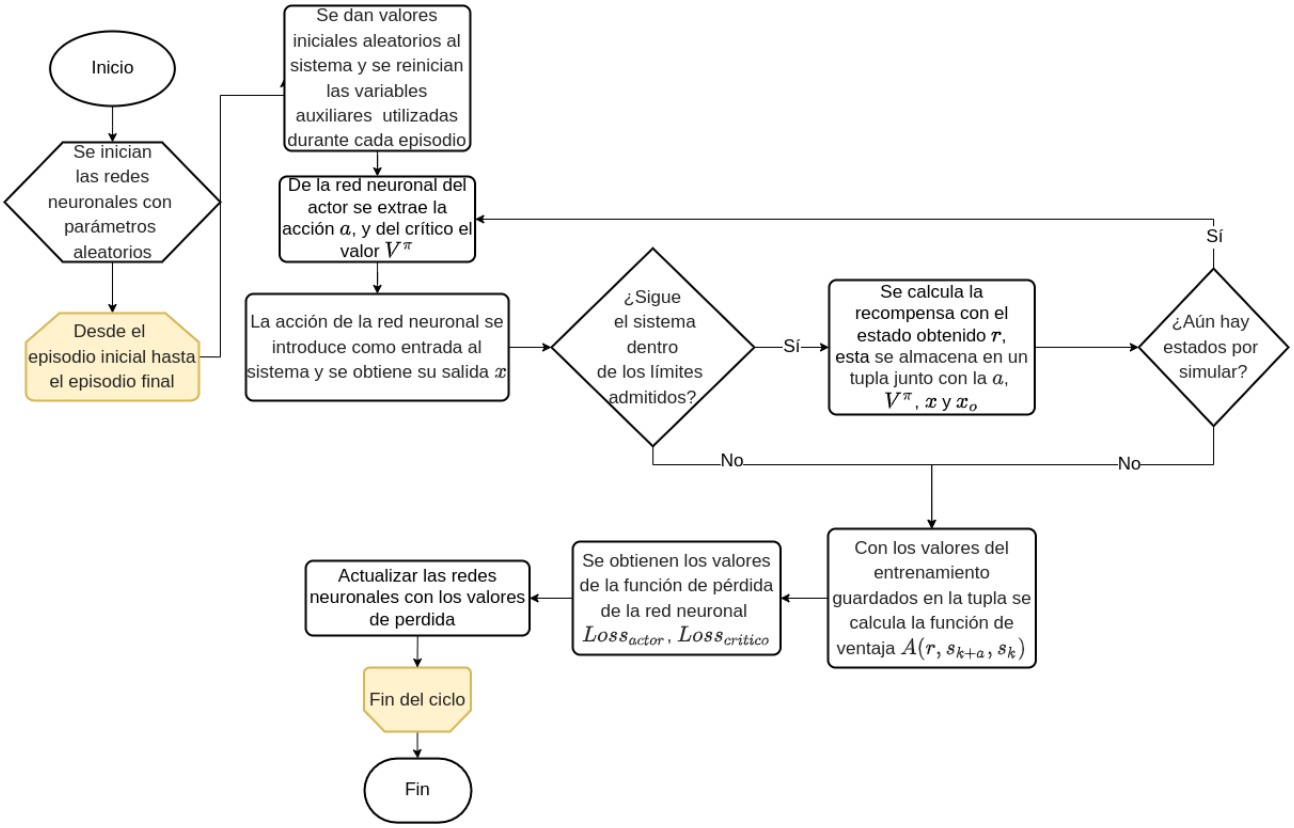

Figura 2.3: Diagrama de flujo del algoritmo actor crítico de ventaja

El diagrama a bloques de cómo interactúan las redes neuronales y la planta se muestra en la Figura 2.4, donde se observa que el bloque del actor es el que aplica la entrada de control u sobre el sistema, de donde por medio de sensores se obtiene el estado de la planta x, con este y la $referencia$ calculamos el $error$ el cual es la información que entra al bloque del actor. El otro proceso donde se utilizan los estados que entregan la planta es en

el cálculo de la función de recompensa, la cual regresa la recompensa r para cada acción, junto a el valor $V^{\pi}(s_k)$ obtenida del bloque del crítico, la cual necesita del error, una vez que se tiene r y $V^{\pi}(s_k)$, se calcula la función de ventaja $A(r, s_{k+a}, s_k)$, para terminar el proceso se realiza la propagación hacia atrás, las respectivas pérdidas para cada red neuronal, ecuaciones (2.2,2.3). Si este diagrama de flujo no es suficiente, se incluye un ejemplo del código en el apéndice A

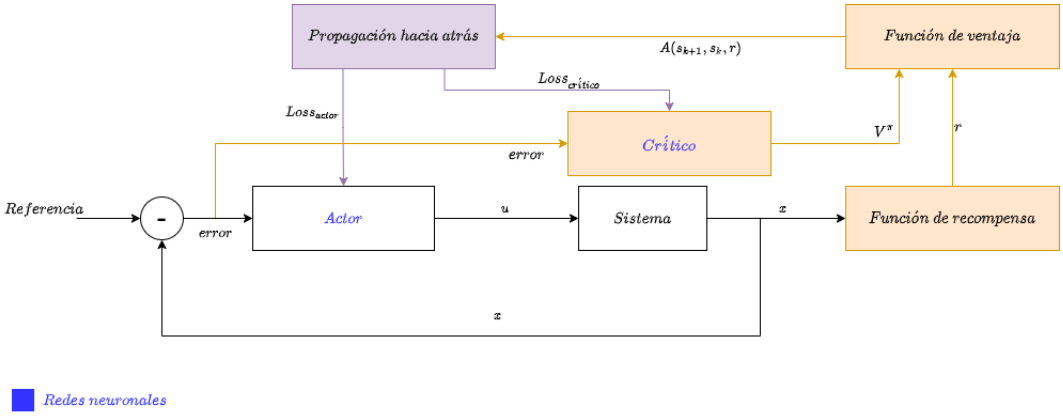

Figura 2.4: Esquema completo del controlador durante el entrenamiento

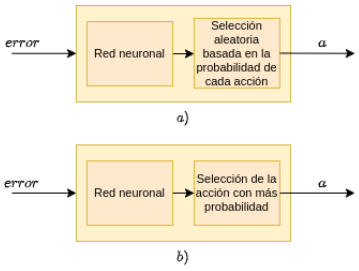

Figura 2.5: Esquema del actor durante el entrenamiento a) y durante las pruebas de desempeño b)

En la Figura 2.4, en color negro se muestran los bloques que son utilizados para probar el controlador, es decir esta conexión en lazo cerrado entre el actor y el sistema es lo único que es necesario para implementar el controlador después del entrenamiento. Aunque el bloque del actor tiene otra modificación, durante el entrenamiento, las salidas de la red

neuronal representan probabilidades y son tomadas de manera aleatoria, pero considerando el peso probabilístico de que pase esta, es decir que todas son posibles aunque la que tenga más probabilidad será elegida mayormente que las demás, esto para favorecer la exploración durante todo momento, esto se muestra en la Figura 2.5.a. Para probar el desempeño se toma siempre la acción que tiene más probabilidad de que pase, es decir las demás acciones no tienen oportunidad de pasar, así el diagrama a bloques de la Figura 2.5.b, muestra esto. Aunque hay que resaltar que esta modificación no fue hecha en la prueba de desempeño de la planta física debido a que se implementó después de los experimentos realizados.

2.3. Funciones de Recompensa

Un parte fundamental del método del actor crítico de ventaja A2C es la función de recompensa, esta premia o castiga las acciones que realiza el controlador, con base en esto se forma el comportamiento de las redes neuronales del crítico y del actor. Esta función es propuesta, aunque en el estado del arte revisado para esta tesis no se encontró algún método para poder establecer la función de recompensa. Hay funciones de recompensa propuestas que funcionan de manera general para cualquier sistema que se desee controlar y hay algunas otras que solamente funcionan para la planta que fue diseñada.

2.3.1. Función de recompensa del estado del arte

Por ejemplo para el control del sistema de péndulo invertido en el simulador Gym es usada la función de recompensa descrita en la ecuación (2.4). Se aclara que el simulador Gym es un paquete para Python desarrollado por el grupo de investigadores de OpenAI para probar algoritmos de aprendizaje reforzado. [Brockman et al., 2016]

$$R(\dot{\alpha}, \alpha, u) = -(\alpha^2 + 0.1\dot{\alpha} + 0.001 \cdot u^2) \tag{2.4}$$

En la ecuación (2.4), donde u, α y $\dot{\alpha}$ son la entrada de control, posición y velocidad para el sistema del péndulo invertido, la recompensa más alta en el simulador Gym es de "0", esta función de recompensa está diseñada para esta planta en específico, donde el objetivo es que el sistema se mantenga en la posición "0", sin velocidad y con una entrada de control también en "0", por lo que esta función de recompensa debe de funcionar para este sistema, sin embargo para esta tesis se utiliza en observar si sistemas parecidos pueden utilizar la misma función de recompensa y para comparar su desempeño contra otras funciones de recompensa.

Otro tipo de funciones de recompensa son las que se propone de manera general para cualquier tipo de sistema que se desee controlar, se muestra en la ecuación (2.5) una función de este tipo. [Spielberg et al., 2017] [Bejar and Moran, 2018]

$$R = \begin{cases} c, & si \quad |X - X_{ref}| \leq \epsilon \\ -|X - X_{ref}|, & de\ lo\ contrario \end{cases} \tag{2.5}$$

En la ecuación (2.5) c y ϵ son constantes positivas que se proponen. c es la recompensa máxima posible y ϵ define el rango de error donde se otorga la máxima recompensa. X es el vector conformado por los estados del sistema $X = [x_1, x_2, \ \dots \ , x_n]$ y X_{ref} es el vector de las referencias para cada uno de estos estados. En la Figura 2.6 se encuentra una representación de la función de recompensa.

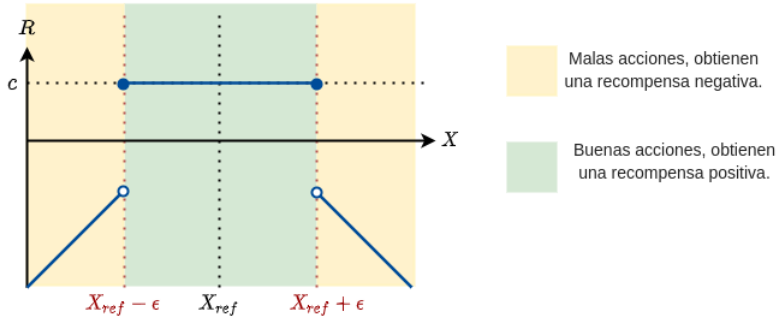

Figura 2.6: Representación gráfica de la función de recompensa de la ec. (2.5)

De la ecuación (2.5), se observa que todo error que este fuera un rango delimitado por ϵ tendrá una recompensa negativa, entre mayor sea el error en el sistema mayor será el número negativo, por el contrario, cuando se está dentro del rango se le da una recompensa positiva, con esto se marca la diferencia entre una buena y mala decisión, el algoritmo maximiza la recompensa por lo que tiende a optar por acciones que otorguen las recompensas positivas. [Spielberg et al., 2017] [Bejar and Moran, 2018]

2.3.2. Función de recompensa propuesta

Se propone una nueva función de recompensa basada en el comportamiento de sistemas lineales, que podría mejorar la evolución del aprendizaje a comparación de las ya utilizadas en el estado del arte.

Para explicar el funcionamiento de la función de recompensa propuesta, se parte de un sistema con dos estados a controlar x_1 y x_2, dos referencias a la cuales denominaremos como x_{ref_1} y x_{ref_2}. La representación del sistema y las referencias en un plano de dos dimensiones se vería como se muestra en la Figura 2.7

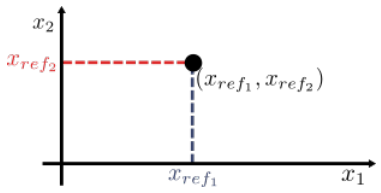

Figura 2.7: Referencia en el plano

Suponiendo que el sistema se encuentre en un estado inicial x_{o_1} y x_{o_2}. A partir de este estado se tomará dos acciones diferentes a_1 y a_2, las cuales tendrán diferentes resultados que llevan al sistema a unos nuevos estados, estos se denominaran como $x_{1_{a_1}}$ y $x_{2_{a_1}}$ para la acción a_1 y los estados $x_{1_{a_2}}$, $x_{2_{a_2}}$ para la acción a_2. Se puede observar una representación gráfica de esto en la Figura 2.8.

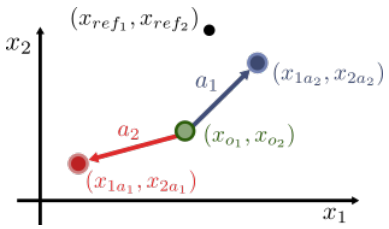

Figura 2.8: Acciones en el plano

Se observa que la acción a_1 es mejor que la acción a_2, debido a que acerca más hacia la referencia. Es decir que todas las acciones que acerquen los estados del sistema a la referencia, son las que se busca premiar, por el contrario, las acciones, como a_2, que no son cercanos a la referencia deben ser castigadas.

El sistema de aprendizaje reforzado siempre busca maximizar la recompensa por lo que busca números positivos y evade los negativos, es por eso que para premiar el comportamiento son usados números positivos y para el castigo negativos. Se ilustra esto en la Figura 2.9 donde la flechas que llevan a la referencia tienen un signo positivo y las que no lo hacen con un signo negativo.

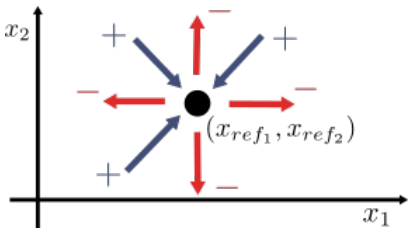

Figura 2.9: Comportamiento deseado en el plano

Este comportamiento está inspirado en la manera en como un punto de equilibrio de un sistema estable atrae todas las trayectorias hacia él. Con esto en mente se propone la función de recompensa descrita en la ecuación (2.6).

$$R = \begin{cases} C, \ si \quad |\sum_{n=0}^{N} c_n(e_{n_{k+1}})| \leq \epsilon \\ \sum_{n=0}^{N} c_n(e_{n_k}) - \sum_{n=0}^{N} c_n(e_{n_{k+1}}), \quad de\ lo\ contrario \end{cases} \tag{2.6}$$

En la ecuación (2.6), N es la cantidad de variables a controlar del sistema, c_n son constantes positivas las cuales ponderan la importancia de cada variable del estado, C es constante y positiva, ϵ es una constante positiva, e_{n_k} es el valor error del estado enésimo antes de aplicar la acción de control, se muestra su fórmula en la ecuación (2.7-a) y $e_{n_{k+1}}$ es el valor absoluto del error después de aplicar la acción tomada por el actor, mostrada en la ecuación (2.7-b).

$$\begin{aligned} &a)\ e_{n_k} = |x_{ref_n} - x_{n_k}| \\ &b)\ e_{n_{k+1}} = |x_{ref_n} - x_{n_{k+1}}| \end{aligned} \tag{2.7}$$

En la ecuación (2.7), x_{ref_n} es la referencia del enésimo estado, x_{n_k} es el enésimo estado antes de que el actor aplique la acción de control y $x_{n_{k+1}}$ es el enésimo estado después

de que el actor aplique la acción de control.

En la función de recompensa (2.6), cuando la magnitud de la suma de los errores después de que el actor aplica la acción de control es mayor, es decir $\sum_{n=0}^{N} c_n(e_{n_k}) < \sum_{n=0}^{N} c_n(e_{n_{k+1}})$, significa que se alejó del sistema y, por lo tanto, la recompensa es negativa, si sucede lo contrario $\sum_{n=0}^{N} c_n(e_{n_k}) > \sum_{n=0}^{N} c_n(e_{n_{k+1}})$, significa que el sistema está más cerca de la referencia y se otorga una recompensa positiva, todo esto siempre cuando $\sum_{n=0}^{N} c_n(e_{n_{k+1}}) > \epsilon$, de lo contrario la recompensa es C debido a que el sistema se encuentra cerca de la referencia.

2.4. Entrenamientos

En esta sección se describe cómo se realizan los entrenamientos, es importante señalar las diferencias que hay entre ellos, aunque sean mínimas y se use el mismo algoritmo, esto ayuda a ver los cambios que hacen al enfrentar un sistema diferente o cuando se implementa físicamente.

2.4.1. Modelos de los sistemas usados

Se comienza por verificar los diferentes modelos usados para cada entrenamiento, ya que se utilizan distintas plantas, con diferente complejidad, estos se simulan para las pruebas.

Motor de CD

El modelo para el motor de CD utilizado es un modelo lineal, el cual se obtiene de [Basilio and Moreira, 2004]. Se selecciona este modelo lineal debido a que en las primeras pruebas se busca comprobar el algoritmo de control con un sistema sencillo. Este modelo propone como variables de estado la velocidad angular del motor ω y la corriente de la armadura i, como se muestra en la ecuación (2.8).

$$\begin{aligned} x_1 &= i \\ x_2 &= \omega \end{aligned} \tag{2.8}$$

Con esto, el espacio de estados del sistema queda como se muestra en ecuación (2.9).

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_{gm}}{L_a} \\ \frac{K_a}{J} & -\frac{f}{J} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} u \tag{2.9}$$

Los valores de cada uno de los parámetros del modelo se definen en la Tabla 2.1, obtenidos de [Basilio and Moreira, 2004]

Tabla 2.1: Tabla parámetros del motor de CD

Parámetro	Valor
Inductancia de armadura L_a	$3.4e^{-3}$
Resistencia de armadura R_a	2.30
Constante de FEM K_{gm}	0.0453
Constante de torque K_a	0.0453
Constante de fricción f	$5.23e^{-5}$
Momento de inercia del motor J	$3.72e^{-5}$

Péndulo rotacional invertido

Otro sistema que se utiliza es el del péndulo rotacional invertido, que se busca implementar de manera física y se encuentra disponible en el laboratorio para su uso.

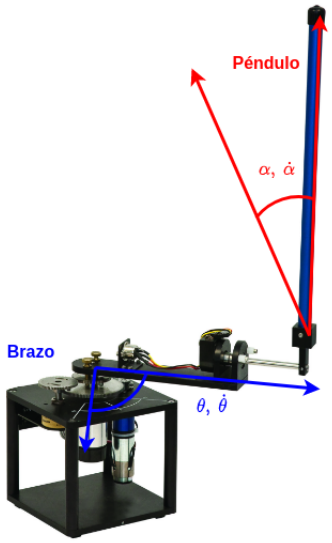

Figura 2.10: Figura del modelo del péndulo rotacional invertido, fuente: www.quanser.com

El modelo del péndulo rotacional invertido usado, es el fabricado por la empresa Quanser. Se utiliza el sistema no lineal y los parámetros que vienen en la documentación que provee la empresa y de los siguientes artículos [Diao, 2016] [Yu, 2018]. Para el modelo se utilizan como variables de estado la posición y velocidad tanto del brazo como del péndulo los cuales se ejemplifican en la Figura 2.10, donde $(\theta, \dot{\theta})$ corresponden a la posición y velocidad angular del brazo, y $(\alpha, \dot{\alpha})$ la posición y velocidad angular del péndulo.

$$
\begin{aligned}
a &= \frac{m_p\ L_r^2}{4} \\
b &= m_p\ L_r^2 + a + J_r \\
c &= \frac{m_p\ L_r\ L_p}{2} \\
d &= J_p + a \\
e &= \frac{m_p\ L_p\ g}{2} \\
\tau &= \frac{n_m\ n_g\ K_g\ K_t\ (V_m - K_m\ K_g\ \dot{\theta})}{R_m}
\end{aligned}
\tag{2.10}
$$

$$\begin{aligned} b - a\ cos(\dot{\alpha})^2\ \ddot{\theta} - (c\ cos(\alpha))\ \ddot{\alpha} + 2a\ sin(\alpha)\ cos(\alpha)\ \dot{\theta}\ \dot{\alpha} + \\ (c\ sin(\alpha))\ \dot{\alpha}^2 + B_r\ \dot{\theta} = \tau \\ -c\ cos(\alpha)\ \ddot{\theta} + d\ \ddot{\alpha} - a\ cos(\alpha)\ sin(\alpha)\ \dot{\theta}^2 - \\ e\ sin(\alpha) + B_p\dot{\alpha} = 0 \end{aligned} \tag{2.11}$$

Para obtener el modelo en espacio de estados y simularlo, de las ecuaciones (2.11) se despeja la aceleración del brazo y la aceleración del péndulo, para después sustituir los valores de las constantes enunciados en la ecuación (2.10), una vez hecho lo anterior se proponen los estados como $x_1 = \theta,\ x_2 = \dot{\theta},\ x_3 = \alpha$ y $x_4 = \dot{\alpha}$, con esto se tiene el espacio de estados, debido a que las ecuaciones son extensas no se colocaron las ecuaciones obtenidas. La entrada del sistema está dada por $u = Vm$. Se coloca la Tabla 2.2 con todos los valores de los parámetros.

Tabla 2.2: Tabla parámetros del péndulo rotacional invertido

Parámetro	Valor
Resistencia de armadura R_m	2.6
Constante del engranaje K_g	70
Constante de Fuerza electromotiz (FEM) K_m	0.007677
Constante de torque K_t	0.007682
Eficiencia de los engranajes n_g	-0.9973
Eficiencia del motor n_m	-0.6144
Longitud del péndulo L_p	0.33655
Inductancia de armadura L_r	0.2159
Masa del péndulo m_p	0.127
Inercia del péndulo J_p	0.001198
Inercia del brazo J_r	$9.98e^{-4}$
Viscosidad péndulo B_p	0.0024
Viscosidad brazo B_r	0.0024
Constante de la gravedad g	9.81

2.4.2. Descripción de los entrenamientos

Los parámetros de las redes neuronales o del mismo entrenamiento que se utilizan en esta tesis se obtienen del proceso de prueba y error, mediante simulaciones y experimentos de laboratorio, encontrar los parámetros tomó gran parte del tiempo, debido a no haber ningún método para su búsqueda en ninguna referencia, esto al menos en las fuentes que se citan.

Se realizan diferentes entrenamientos para ver el comportamiento de control en varias situaciones, dos se realizan en simulación y una de manera práctica, en simulación las pruebas se realizan con dos modelos, el de un motor y el del péndulo rotacional invertido, este último también es utilizado para la prueba práctica, aunque se hacen cambios para el

entrenamiento de la prueba práctica, estos cambios son mencionados más adelante.

Las redes neuronales del actor como del crítico, tienen como diferencia el número de salidas lo demás es idéntico. Estas redes fueron implementadas por medio del paquete PyTorch, el cual permite implementar redes neuronales en Python, otra característica importante es que permite la aceleración de ejecución de las redes neuronales utilizando una tarjeta de video. Dentro de las conexiones entre capas disponibles en PyTorch, se seleccionó la lineal, que en la documentación es descrita como una transformación lineal obtenida de la ecuación $y = A^T + b$, donde y es la salida, x la entrada, A los pesos o coeficientes por los cuales se multiplica y b es el sesgo o coeficientes los cuales son sumados.

Adicional a la arquitectura, se seleccionaron como función de activación en la capa oculta $HardTan()$ en ambas redes neuronales. El actor tiene en la capa de salida una función de activación $SoftMax()$, el crítico no cuenta con función de activación a la salida esta salida.

La arquitectura queda tal cual se muestra en las Figuras (2.1, 2.2), esta red neuronal se clasificaría como multicapa totalmente conectada, lo que se conoce como red neuronal pre alimentada. La razón de escoger esta red neuronal simple y la función activación lineal es porque cuando se realizaron las pruebas en el laboratorio con el péndulo rotacional invertido esta red neuronal se ejecutaba más rápido que las demás, por lo que es posible hacer la retroalimentación de los parámetros con la planta física.

Ambas redes neuronales tienen como entrada los errores de cada una de las variables de estado del sistema, otra similitud es la cantidad de neuronas en la capa oculta que se optó por 25, esto no cambia en ninguna prueba, a este número se llego mediante la experiencia obtenida en la realización de pruebas. La red neuronal del actor tiene varias salidas, cada una de estas representa una acción a tomar las cuales son: aumentar, disminuir o mantener igual la señal de control. En el caso de aumentar y disminuir es un delta Δu definido en el inicio del entrenamiento.

Tabla 2.3: Tabla comparativa de entrenamientos

Prueba	Número de entradas y salidas de la red neuronal del actor	Número de entradas y salidas de la red neuronal del crítico	Paramentos del entrenamiento	Función de recompensa	Parámetros de la funciones de recompensa
Simulación del modelo lineal del motor de CD	Entradas = 1 Salidas = 3	Entradas = 1 Salidas = 1	$lr \rightarrow 3e^{-7}$ $episodios \rightarrow 1000$ $\gamma \rightarrow 0.99$ $t_f \rightarrow 20$ $\Delta t \rightarrow 10e^{-3}$ $\Delta u \rightarrow 0.1$	Ecuación (2.6), con $x_1 \rightarrow \omega$	$c_1 = 20$ $\epsilon = 10$ $C = 100$
Simulación del modelo no lineal péndulo rotacional invertido	Entradas = 4 Salidas = 3	Entradas = 4 Salidas = 1	$lr \rightarrow 5e^{-7}$ $episodios \rightarrow 4000$ $\gamma \rightarrow 0.99$ $t_f \rightarrow 20$ $\Delta t \rightarrow 10e^{-3}$ $\Delta u \rightarrow 0.5$	Ecuación (2.4)	
				Ecuación (2.5), con $x_1 \rightarrow \theta$ $x_2 \rightarrow \alpha$	$c = 100$ $\epsilon = 0.1$
				Ecuación (2.6), con $x_1 \rightarrow \theta$ $x_2 \rightarrow \dot{\theta}$ $x_3 \rightarrow \alpha$ $x_4 \rightarrow \dot{\alpha}$	$c_1 = 50$ $c_2 = 1$ $c_3 = 100$ $c_4 = 1$ $\epsilon = 100$ $C = 100$
Prueba física del péndulo rotacional invertido	Entradas = 4 Salidas = 11	Entradas = 4 Salidas = 1	$lr \rightarrow 5.5e^{-7}$ $episodios \rightarrow 600$ $\gamma \rightarrow 0.99$ $t_f \rightarrow 20$ $\Delta t \rightarrow 20e^{-3}$ $\Delta u \rightarrow 0.5$	Ecuación (2.4)	
				Ecuación (2.5), con $x_1 \rightarrow \theta$ $x_2 \rightarrow \alpha$	$c = 1000$ $\epsilon = 0.1$
				Ecuación (2.6), con $x_1 \rightarrow \theta$ $x_2 \rightarrow \dot{\theta}$ $x_3 \rightarrow \alpha$ $x_4 \rightarrow \dot{\alpha}$	$c_1 = 50$ $c_2 = 1$ $c_3 = 100$ $c_4 = 1$ $\epsilon = 100$ $C = 1000$

Se presenta la Tabla 2.3, en donde se muestran los parámetros de cada entrenamiento. En la primera columna se describe la planta utilizada,así como si se realiza una simulación o implementación física. En la segunda y tercera columna, se colocan las entradas y salidas utilizadas en las redes neuronales tanto del actor como del crítico, remarcando que no se incluye la capa oculta, debido a que en todas las pruebas se usaron 25 neuronas, y las entradas de las redes son los errores relacionados con cada uno de los estados del sistema. En la cuarta columna se expone todos los parámetros necesarios para: la retroalimentación de las redes neuronales, número de episodios, cambios de la entrada u y la duración máxima de cada episodio t_f, junto con el paso de simulación o en el caso de implementación física

el tiempo de muestreo, denotado como Δt. En la quinta columna se muestra la función de recompensa utilizada y cómo es que se asignaron los estados del sistema a esta. En la última columna se presentan los parámetros constantes utilizados para las funciones de recompensa.

En los entrenamientos se busca que las circunstancias sean lo más parecidas posibles, por eso los coeficientes iniciales en las redes neuronales, las condiciones iniciales y las referencias durante el entrenamiento son iguales para las simulaciones, incluso si se cambia de función de recompensa, aunque esto no se logra en la práctica debido a que algunos de estos parámetros no pueden ser cambiados. Otras características a mencionar es que, las simulaciones se realizan en Python por medio del método Runge Kutta de cuarto orden, con un paso de simulación variable, este es diferente al tiempo de ejecución del algoritmo Δt.

Existen dos formas en las que se termina un episodio, la primera es debido a que se llega al tiempo final, y la segunda es porque los estados superan algunos límites propuestos, esto debido a que en caso del péndulo rotacional invertido se podía llegar a un estado que lleve al sistema de regreso a la referencia; otro motivo es que cuando se realiza la prueba práctica, estos ayudan a que la planta no sufra daños durante el entrenamiento. Los límites para el sistema de péndulo rotacional invertido son $|x_1| < \frac{\pi}{1.5},\ |x_3| < 0.8\pi$, para la simulación del motor de CD no es necesario utilizar límites.

2.5. Conclusiones del Capítulo

En este capítulo se describió como es algoritmo A2C, el cual está compuesto por dos redes neuronales, el actor el cual es el que toma las decisiones, en este caso es el encargado de decidir la entrada de control aplicada al sistema, y la red neuronal llamada crítico que evalúa cual es la mejor acción para el estado dado mediante la exploración. Las funciones de pérdida, ecuaciones (2.2, 2.3), fueron utilizadas para hacer retroalimentación a las redes neuronales para mejorar su desempeño utilizando la función de ventaja. También se incluyó un diagrama de flujo del algoritmo A2C y una explicación de cómo fue utilizado.

Se notó que parte fundamental del algoritmo es la función de recompensa utilizada, debido a esto es que se dedicó una sub sección completa a esta, donde se explica cómo es que estas impactan en el entrenamiento, diferenciando las buenas acciones de las malas, y como el algoritmo buscar maximizar la recompensa obtenida al paso de los episodios, buscando recompensas positivas asociadas a buenas acciones, por lo que se termina reforzando la toma de las decisiones positivas dadas por las funciones de recompensa.

Se describió de manera completa por medio de un ejemplo de cómo es que se llegó a desarrollar la función de recompensa desde cero, como se utilizó de inspiración los sistema lineales para esto, también de la herramienta matemática necesaria para plasmar esta función.

Finalmente se describió como es que se realizaron los entrenamientos para cada sistema, porque aunque se utiliza el mismo algoritmo, este se debe de adaptar al número de estados a controlar que tiene la planta, así como las constantes de la funciones de recompensa, que varían de un sistema a otro. Todos estos detalles fueron descritos en este capítulo.

Capítulo 3

Resultados

3.1. Introducción

En este capítulo se presentan los resultados de las simulaciones y la implementación física que se desarrolla para esta tesis, así como el análisis de estos. Para comparar el desempeño se utilizaran tres diferentes señales de referencia para cada uno de los controladores, los resultados se analizan de manera cuantitativa a lo largo del capítulo.

3.2. Formato de Presentación de las Pruebas

La manera en que se presentan las pruebas en esta tesis es la siguiente: primero se realiza la presentación de la evolución de la recompensa, la cual indica que tan exitoso es el entrenamiento, después se presentan las pruebas de seguimiento de referencia hechas con el último modelo obtenido del entrenamiento, es decir no estará aprendiendo el controlador mientras se hacen las pruebas del mismo, esto debido a practicidad de implementación.

La forma de conocer si un entrenamiento es bueno, es si este llega a su máximo valor de recompensa posible, el cual se puede calcular multiplicando el valor máximo posible de la función de recompensa por el número total de acciones tomadas en el episodio $r_{epi_{max}} = r_{max}\ a_{totales}$. El número de acciones tomadas se puede obtener con una división del

tiempo por episodio entre el periodo de tiempo en el que se ejecuta el algoritmo $a_{totales} = \frac{t_f}{\Delta t}$.

Las gráficas de evolución de recompensa siempre dirigen la recompensa tal cual de cada episodio acompañada de una **recompensa suavizada**, que es calculada promediando cada diez valores de la recompensa.

3.3. Control de un Motor de CD

El primer sistema a presentar es el motor de CD del cual se presenta solamente su simulación, utilizando el modelo lineal, por lo tanto, se le considera la planta más sencilla presentada en esta tesis.

3.3.1. Resultado del entrenamiento

A continuación se presenta la evolución de su aprendizaje, Figura 3.1.

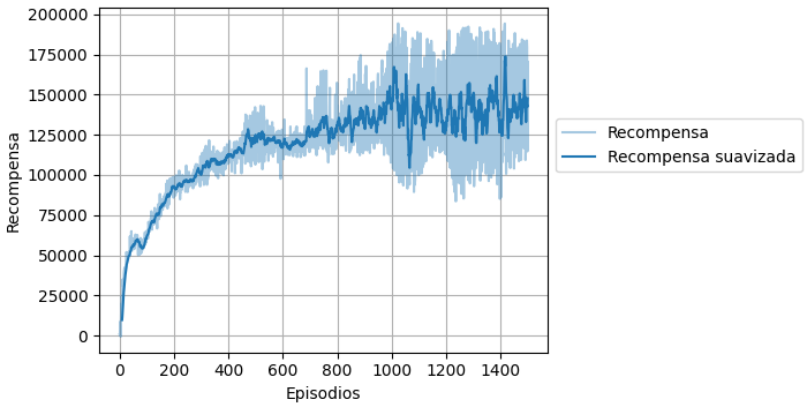

Figura 3.1: Evolución de la recompensa del motor de CD

Se puede calcular el valor máximo para esta prueba con los datos de la Tabla 2.3, lo que indica que $r_{epi_{max}} = 200,000$, se observa como la recompensa en algunos episodios logran acercarse hasta esta recompensa máxima, pero la recompensas suavizada llega cerca

de los 150, 000, así que este no es un resultado completamente exitoso, pero si es el mejor obtenido durante las pruebas. En los primeros episodios la recompensa llega a su valor máximo, pero después de los 800 episodios parece ya no tener crecimiento.

3.3.2. Prueba del controlador

El controlador es probado primeramente con una referencia constante, obteniendo buenos resultados aunque con error en estado estable. En la Figura 3.2 se puede observar como cambio el estado a través del tiempo, la prueba tiene una duración de 60 s, llegando a la referencia en alrededor de 1.5 s. Se realiza un acercamiento a la gráfica para observar como existe un error en estado estable. Se presenta un histograma del error durante la prueba en la Figura 3.3, donde efectivamente la media del error, colocada al pie de la gráfica, muestra que se tiene un error en estado estable $0.2\frac{Rad}{s}$.

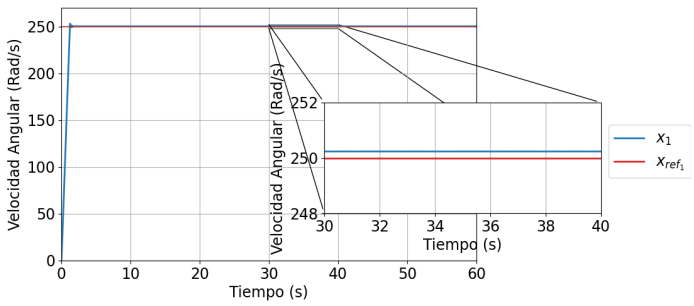

Figura 3.2: Referencia constante, motor de CD

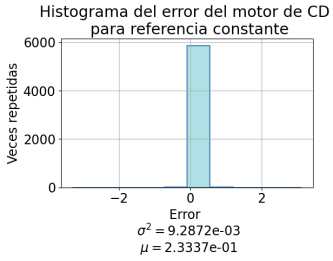

Figura 3.3: Histograma de referencia constante, motor de CD

La siguiente prueba, Figura 3.4, se realiza con múltiples referencias constantes que cambian. En esta prueba todas las referencias son alcanzadas, de nuevo se realiza un acercamiento, en donde se puede observar el error en estado estable. Al igual que la prueba anterior se realizó un histograma del error, que se muestra en la Figura 3.5, aunque para este caso el error en estado estable no es evidente, esto debido a que los errores entre los cambios de referencia modifican el sesgo.

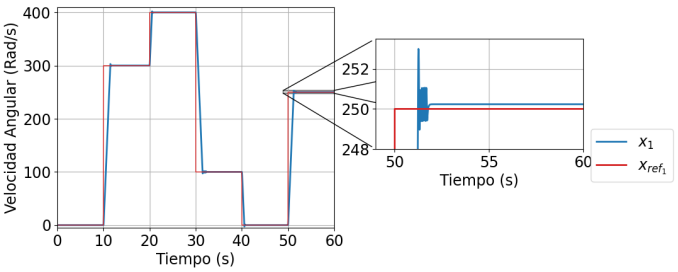

Figura 3.4: Múltiples constantes, motor de CD

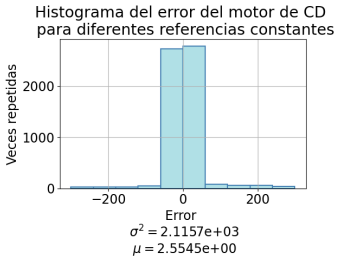

Figura 3.5: Histograma de múltiples constantes, motor de CD

La última prueba realizada a este controlador se realiza utilizando una señal senoidal 200 $sin(t)$, en la cual se basa en la gráfica de la Figura 3.6. Se puede comentar que tiene buen desempeño siguiendo esta referencia, se muestra un histograma del error de esta prueba, Figura 3.7 , el cual no contiene error en estado estable alguno esto tal vez por la naturaleza de la señal de referencia.

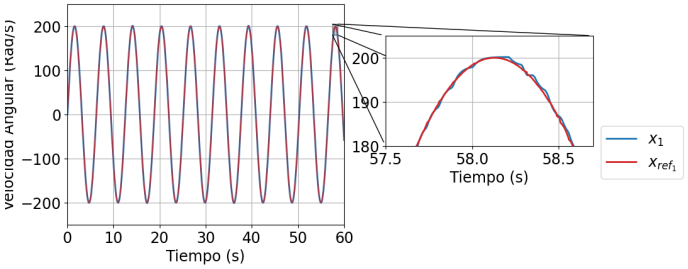

Figura 3.6: Referencia variante con el tiempo, motor de CD

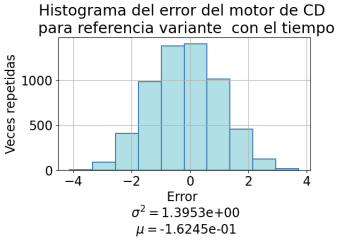

Figura 3.7: Histograma de referencia variante con el tiempo, motor de CD

3.4. Péndulo Rotacional Invertido

En esta subsección se enfoca en el péndulo rotacional invertido, se presentan los resultados con el modelo no lineal simulado, a diferencia de la prueba del motor donde se utiliza una función de recompensa. En esta se utiliza las tres funciones de recompensa mencionadas en el capítulo anterior, por lo que se comparan el desempeño de cada una.

3.4.1. Resultados de los entrenamientos

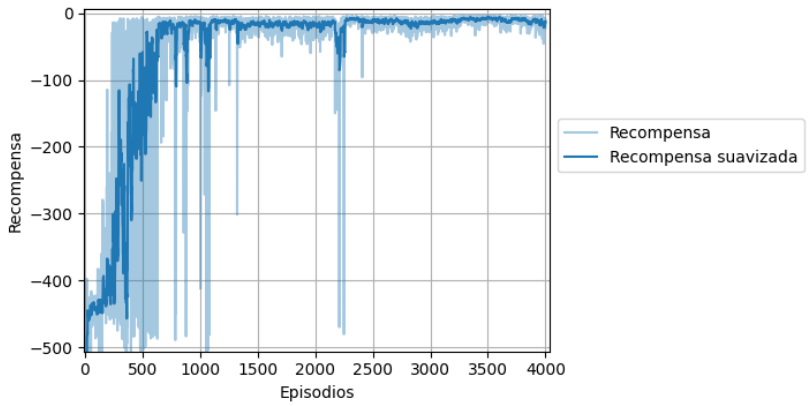

Figura 3.8: Evolución de la recompensa con la ecuación (2.4) en la simulación

La primera evolución de recompensa que se muestra es la de la función de recompensa de la ecuación (2.4), esto en la Figura 3.8, donde la recompensa máxima por episodio es de $r_{epi_{max}} = 0$, este valor máximo no es alcanzado en el entrenamiento.

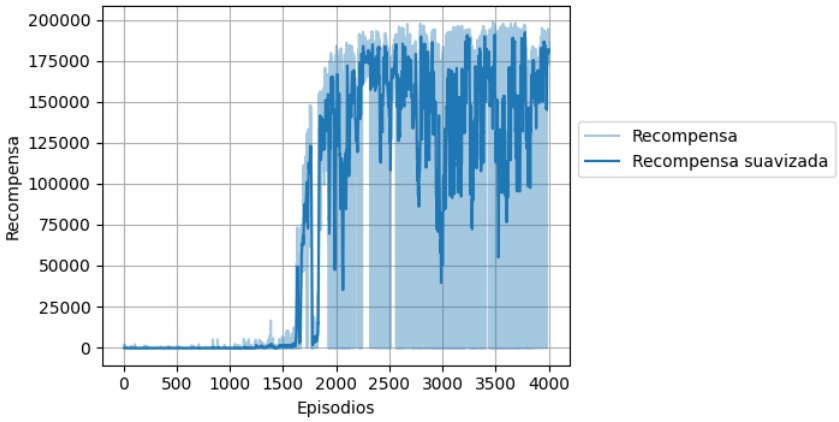

Figura 3.9: Evolución de la recompensa con la ecuación (2.5) en la simulación

La evolución de la recompensa durante el entrenamiento con la ecuación (2.5) se observa en la Figura 3.9. La recompensa máxima por episodio es $r_{epi_{max}} = 200,000$, y en algunos instantes se aproxima en su valor máximo, se presenta también, que la recompensa suavizada tiene variaciones, y esta alcanza hasta $175,000$.

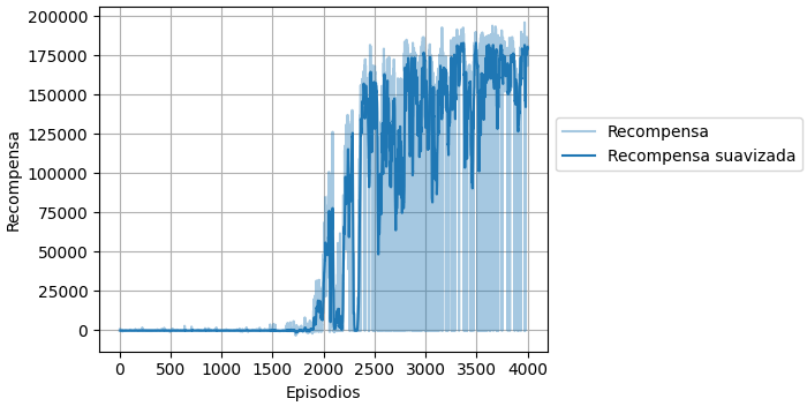

Figura 3.10: Evolución de la recompensa con la ecuación (2.6) en la simulación

La evolución de la recompensa de la ecuación (2.6), mostrada en la Figura 3.10, tiene un valor máximo de recompensa de $r_{epi_{max}} = 200,000$, este no es alcanzado durante el entrenamiento, se aproxima hasta $175,000$, comparado con los entrenamientos anteriores este mantuvo bajos valores durante más episodios aproximadamente $2,000$.

3.4.2. Prueba de los controladores entrenados con las diferentes funciones de recompensa

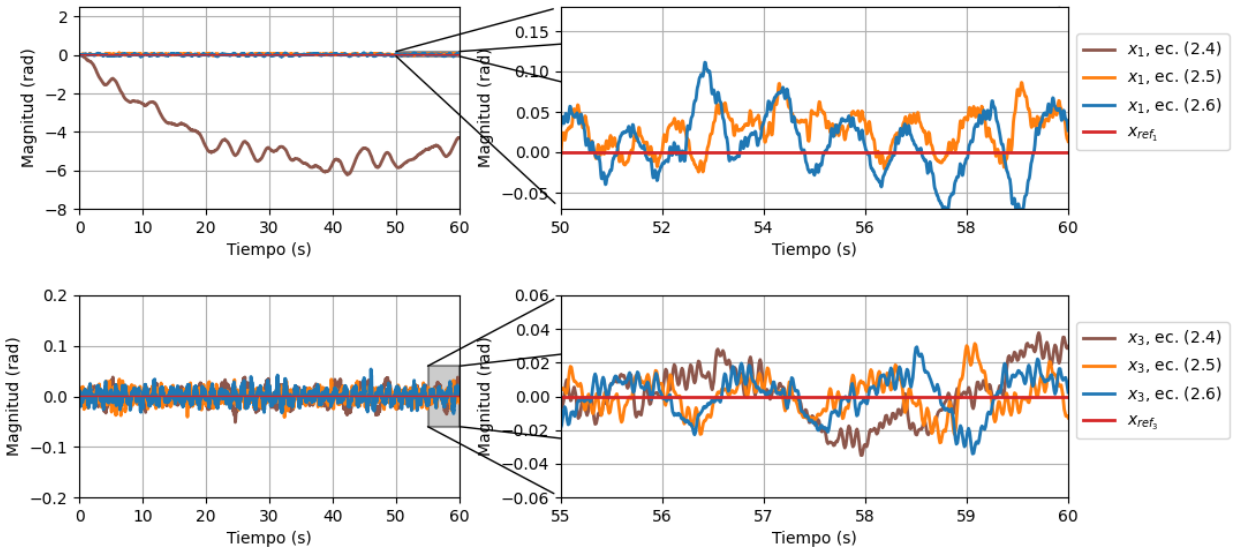

Figura 3.11: Referencia constante péndulo rotacional invertido

Las pruebas a presentadas a continuación se realizan de la misma manera. La primera consiste en seguir una referencia constante en los estados $x_1 = x_3 = 0$, Figura 3.11, donde en la primera gráfica se observa como el controlador entrenado con la ecuación (2.4) es incapaz de seguir la referencia del estado x_1, debido a que no se toma en cuenta este estado en la recompensa, aunque sí puede seguir la referencia en el estado x_3. Por otro lado los controladores entrenados con las ecuaciones (2.5,2.6), si pueden seguir la referencias en los dos estados, aunque los tres controladores presentan oscilaciones alcanzando la referencia.

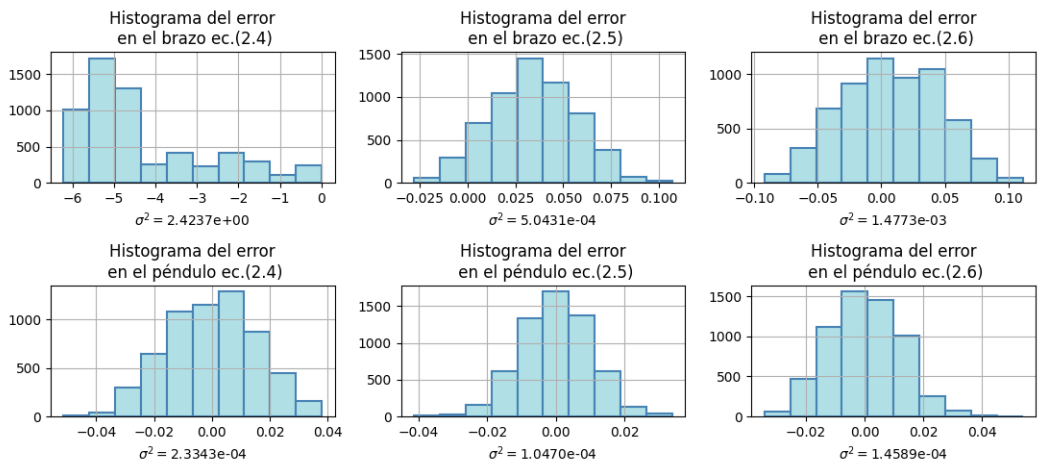

Figura 3.12: Histograma de referencia constante péndulo rotacional invertido

Para comparar cuantitativamente el desempeño de las funciones de recompensa, se realizan los histogramas del error de cada uno de ellos en las dos variables de estado, estos se muestran en la Figura (3.12). Al pie de la gráfica se incluye el cálculo de la varianza como medida de dispersión, en cuanto más bajo sea, mejor desempeño tiene, debido a que hay menos oscilaciones. El controlador con mejor desempeño es el entrenado con la ecuación (2.5), seguido del entrenado con la ecuación (2.6), esto basado en la varianza de los datos.

Se realiza una prueba de seguimiento de múltiples referencias constantes para el estado x_1, Figura 3.13, donde los resultados son muy parecidos a los de la prueba anterior, el control entrenado con la ecuación (2.4) no puede seguir el cambio de referencias, pero si mantiene el estado x_3 en la referencia, mientras que los controladores entrenados con las ecuaciones (2.5,2.6) si siguen la referencia aunque con oscilaciones, obteniendo un resultado bastante parecido entre ellas.

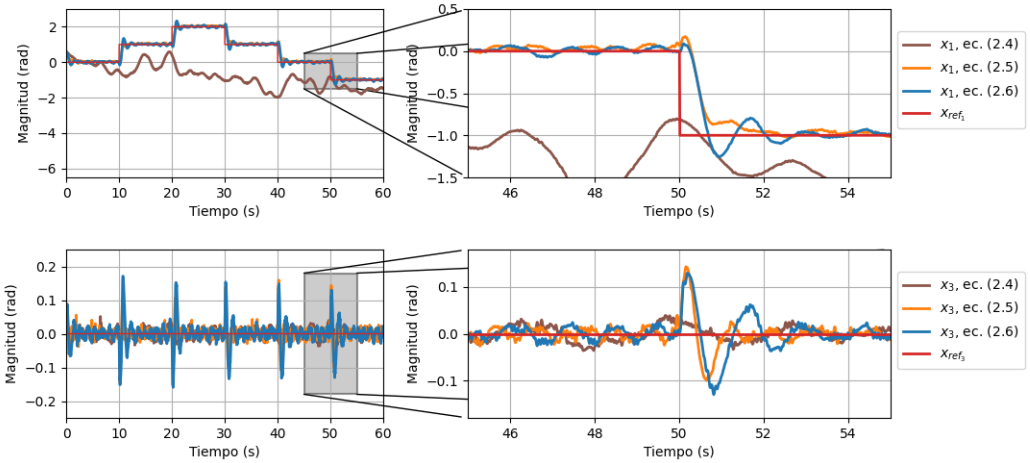

Figura 3.13: Múltiples constantes péndulo rotacional invertido

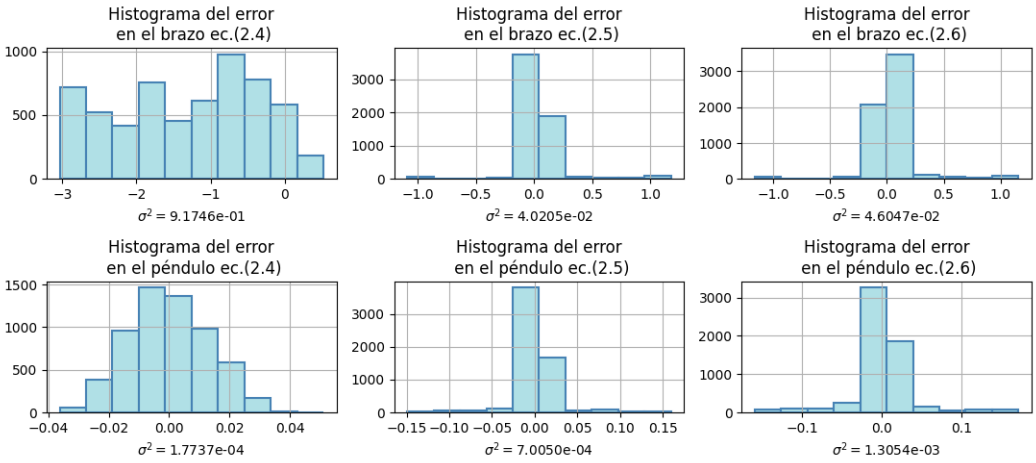

Figura 3.14: Histograma de múltiples constantes péndulo rotacional invertido

El histograma de la prueba de múltiples referencias constantes se observa en la Figura 3.14, donde de nuevo el mejor controlador es el entrenado con la ecuación (2.5).

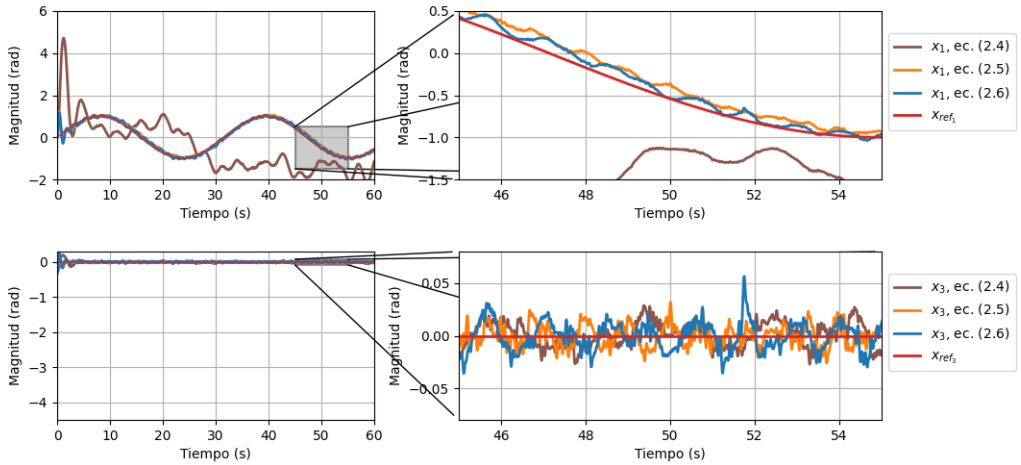

Figura 3.15: Referencia variante con el tiempo péndulo rotacional invertido

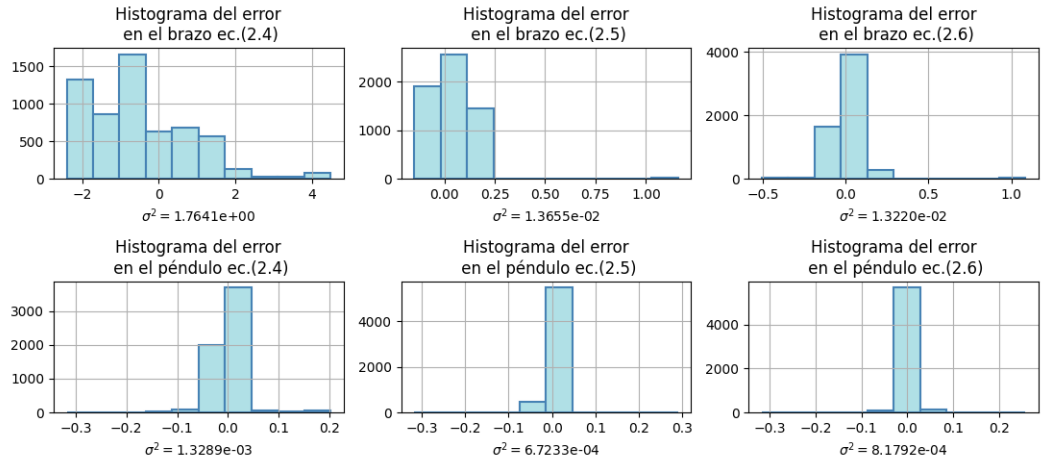

Figura 3.16: Histograma de referencia variante con el tiempo péndulo rotacional invertido

La última prueba se realiza con una señal de referencia variante con el tiempo $sin(0.2 * t)$, Figura 3.15, donde se observa lo mismo que en las pruebas anteriores. El histograma de esta prueba se muestra en la Figura 3.16 donde se tiene un ligero cambio, ya que para el estado x_1 el que tiene mejor desempeño el entrenado por la ecuación (2.6), no así el de la ecuación (2.5) como en las anteriores, aunque este sigue teniendo mejor desempeño en el estado x_3.

3.5. Prueba en Planta Física

En esta subsección se presentan los resultados obtenidos al implementar el algoritmo Advantage Actor Critic (A2C) para el control del péndulo rotacional invertido, adicionalmente se describe bajo qué condiciones se implementó, junto con los resultados obtenidos de los entrenamientos con las diferentes funciones de recompensa.

3.5.1. Implementación

El péndulo giratorio invertido es fabricado por la empresa Quanser. El cual está conformado por el mecanismo del péndulo giratorio invertido, que es la parte mecánica junto el motor eléctrico y sensores, como se muestra en la Figura 3.17 de color azul, la otra parte que es necesaria para utilizar el péndulo es la tarjeta de adquisición de datos, que es el puente entre la computadora el péndulo, ya que se encarga de controlar el motor y leer los sensores a la vez que transfiere esta información a la computadora , está también es fabricada por Quanser. La placa de adquisición de datos se denota de color rojo en la Figura 3.17. La computadora utilizada cuenta con un procesador de cuatro núcleos y 8GB de memoria RAM, marcada de color amarillo en la Figura 3.17.

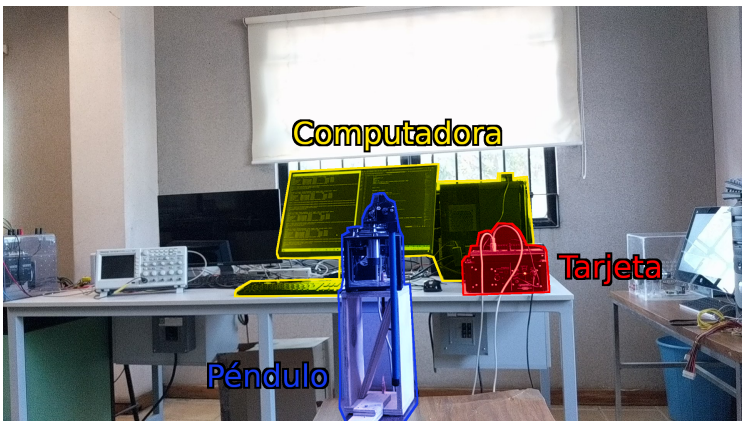

Figura 3.17: Fotografía del experimento

El software utilizado varía en comparación con las simulaciones, debido a que para usar la tarjeta de adquisición de datos se necesitan utilizar controladores proporcionados por la empresa Quanser que no están disponibles en Python solamente en C. La solución que se encontró para este problema fue ejecutar dos programas en paralelo, uno en C que contiene los controladores de la placa y otro en Python con el algoritmo A2C, los cuales comparten información por medio de una conexión interna usando el protocolo UDP. Este enfoque puede ser llamado como micro servicios, donde cada programa hace una parte del proceso, pero con el mismo objetivo. Se muestra un diagrama de la ejecución en la Figura 3.18. Otro cambio sustancial en comparación a las simulaciones es que las redes neuronales son ejecutadas por medio de la tarjeta de video con la que cuenta la computadora, esto para acelerar el proceso y alcanzar el tiempo mencionado en la Tabla 2.3.

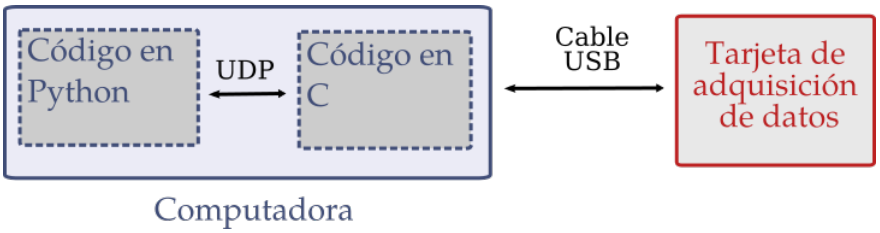

Figura 3.18: Diagrama a bloques del experimento

Para la implementación del controlador en la prueba con la planta física, se cambian características del entrenamiento para su funcionamiento, las cuales son:

- Se duplica el tiempo de simulación o muestreo, debido a que se encuentran limitaciones en el uso de la tarjeta de adquisición de datos.
- La cantidad de acciones de la red neuronal, esto debido a que una vez implementada la prueba física, tener solamente un cambio de la acción del control Δu no es suficiente, porque la señal de control no logra mantener al sistema en la referencia.
- Se cambia el coeficiente de aprendizaje lr, aumentando su valor con el fin de que el aprendizaje sea rápido.

- El número de episodios, se reduce debido a que se observa que con las configuraciones realizadas, el aprendizaje presenta subidas rápidas en la evolución del aprendizaje.

- La forma en que se inicializa el sistema es de forma manual, colocando el péndulo con la mano dentro de una zona programada.

- Debido a que el péndulo no mide la velocidad angular, esta es para calculada por medio del cambio de posición angular dividida entre el cambio de tiempo.

Se deben de mencionar que hay otras dos diferencias con respecto a las simulaciones, estas surgieron como consecuencia de la implementación, y surgieron como elementos no deseados, que afectan el entrenamiento de los controladores. Lo anterior mencionado se describe a continuación:

- El primero surge en relación con la colocación del péndulo en su posición inicial. En la simulación se genera un número aleatorio en un rango seleccionado, lo que hace que se tenga una distribución uniforme de las condiciones iniciales, como se muestra en la Figura 3.19, pero esto no pasa de la misma manera en la prueba física, ya que en esta se tiene que colocar el péndulo con la mano en la condición inicial, que al igual que las simulaciones tiene un rango admisible, al colocar el péndulo con la mano inmediatamente que entra dentro rango admisible el péndulo comienza a actuar el controlador para su entrenamiento, esto ocasiona que se concentre la distribución en las fronteras que es donde inicia el rango admisible, como lo muestra la Figura 3.20.

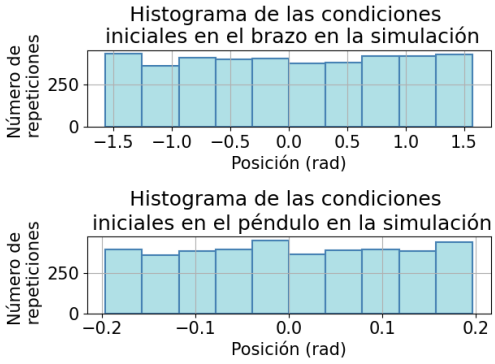

Figura 3.19: Histograma de las condiciones iniciales en la simulación

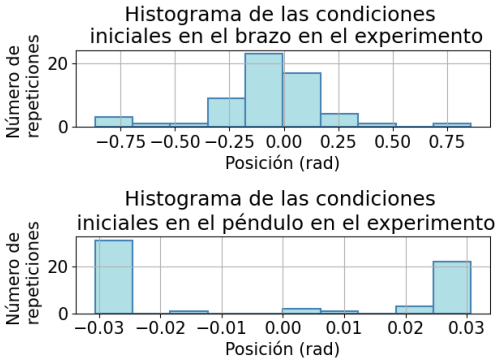

Figura 3.20: Histograma de las condiciones iniciales en la prueba física

- El segundo cambio sucede dentro del software. La ejecución del programa en Python, se mide por medio del reloj de la computadora (tiempo que pasa entre cada muestra), debido a que el tiempo se entrega en un formato que contiene información en horas, minutos y segundos, se requiere ajustar mediante una conversión a que solamente quede en segundos, en este proceso se efectúa de manera correcta pero solamente tomando en cuenta dos decimales, perdiendo así gran resolución. No se pierde información durante el entrenamiento, lo que se ve afectado es la velocidad del péndulo ya que este tiempo es utilizado para este, por lo que la red neuronal no estaría recibiendo de manera correcta la velocidad. Debido a que este desperfecto encontrado al procesar la información y rehacer los experimentos llevaría meses, solamente se informa sobre este para tenerse en cuenta a la hora de analizar los resultados presentes. En la Figura 3.21, se observa cómo es que el tiempo calculado por la computadora solamente oscila entre 0.01 s y 0.02 s, sin valores intermedios entré estos dos, valores como 0.019 s, 0.0198 s, etc son tomados como 0.01 s, debido a que se trunca al segundo decimal, la afectación de este error es bastante, para poner un ejemplo de la muestra con la que se hizo la gráfica de la Figura 3.21 el 37.8 % de los valores son 0.01 s, que es la mitad del tiempo promedio en el que se reciben los datos, dando así una velocidad con la mitad de la magnitud real.

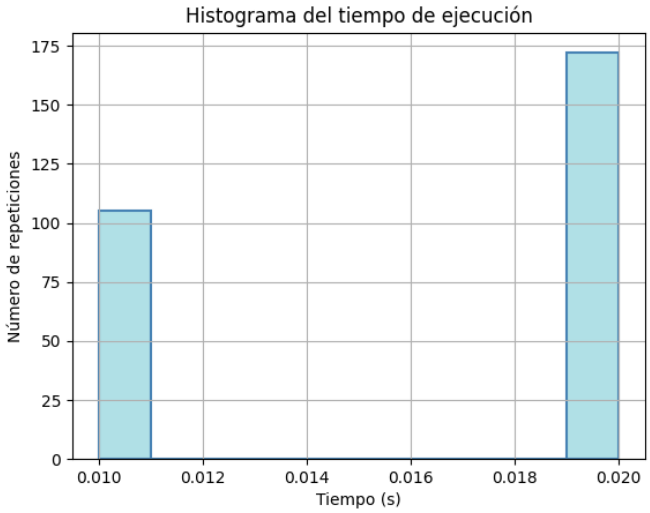

Figura 3.21: Histograma del tiempo de ejecución del algoritmo en la prueba física

3.5.2. Resultados del entrenamiento

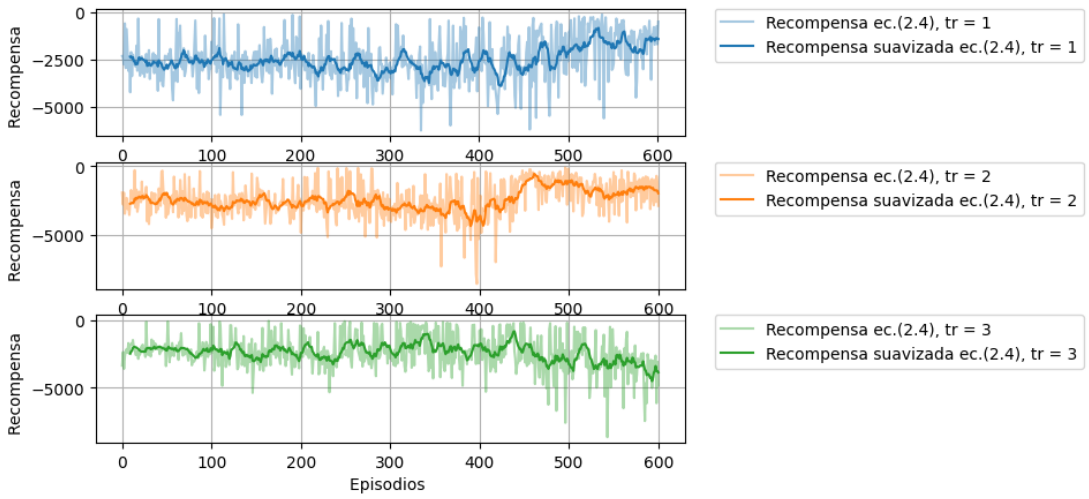

Figura 3.22: Evoluciones de la recompensa con la ecuación (2.4) en la prueba física

De los entrenamientos realizados con la ecuación (2.4), ninguno alcanzó la recompensa máxima por episodio $r_{epi_{max}} = 0$, se muestran los resultados en la Figura 3.22, el

segundo entrenamiento tiene el mejor desempeño. En el episodio 400 asciende el aprendizaje, aunque este disminuye en el episodio 500. El controlador no puede mantener el péndulo de manera vertical.

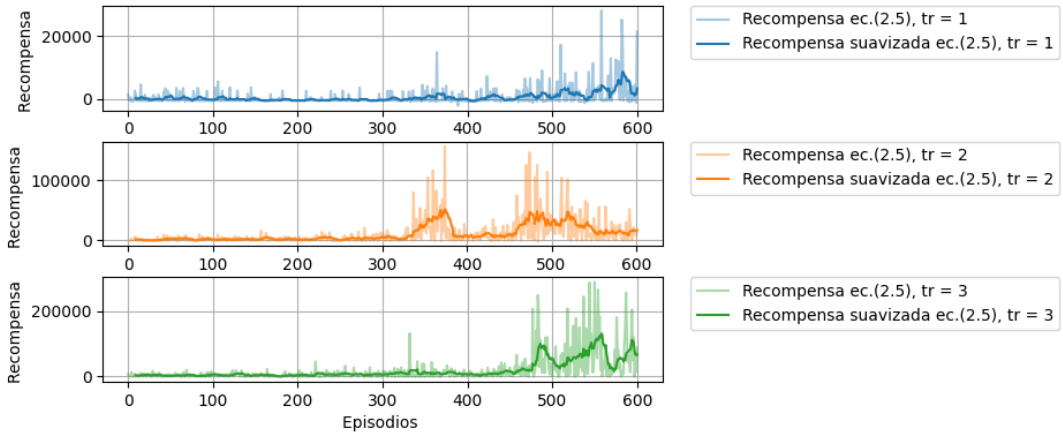

Figura 3.23: Evoluciones de la recompensa con la ecuación (2.5) en la prueba física

Se realizan tres pruebas con la ecuación (2.5), y no se alcanza su máxima recompensa por episodio $r_{epi_{max}} = 1,000,000$. La evolución de la recompensa tiene fluctuaciones, esto se observa en la Figura 3.23, el que tiene menor desempeño es el del tercer entrenamiento. Este controlador no logra mantener el péndulo de manera vertical.

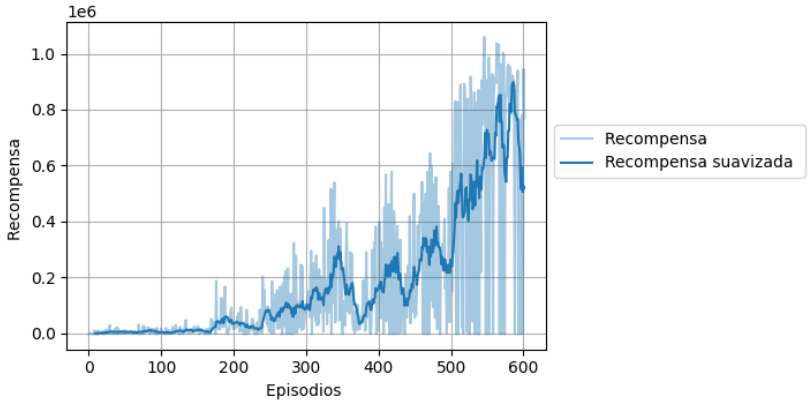

Figura 3.24: Evolución de la recompensa con la ecuación (2.6) en la prueba física

Para el controlador entrenado con la ecuación (2.6), se realizan dos entrenamientos, solamente uno tuvo un desempeño exitoso, y se muestra en la Figura 3.24, donde se observa que se alcanza la recompensa máxima por episodio en $r_{epi_{max}} = 1,000,000$, inclusive es rebasada. Lo anterior debido a los errores en el cálculo de la velocidad, la recompensa suavizada no alcanza al máximo valor. Este es el único entrenamiento exitoso de las pruebas con la planta física, llegando a mantener el péndulo en posición vertical.

3.5.3. Prueba del controlador entrenado con la función de recompensa de la ecuación (2.6)

Solamente al controlador que evolucionó de manera adecuada se le realizan pruebas. La primera prueba es la de seguimiento de una referencia constante, la cual es mostrada en la Figura 3.25, donde se observa como sigue la referencia el controlador con algunas oscilaciones, hay un aumento de las oscilaciones en los primeros $5s$ de la prueba. Se presenta un histograma de esta prueba en la Figura 3.26, si se compara con los resultados de la simulación es parecido.

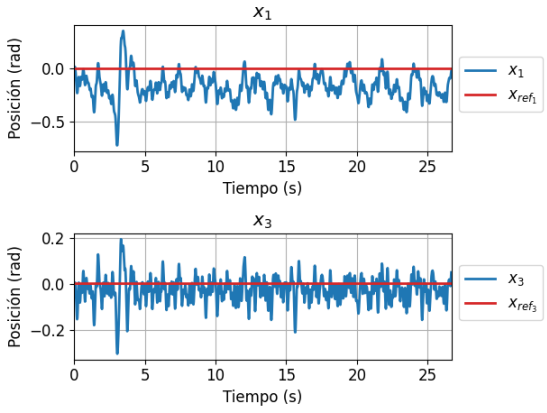

Figura 3.25: Referencia constante en la prueba práctica con la ecuación (2.6)

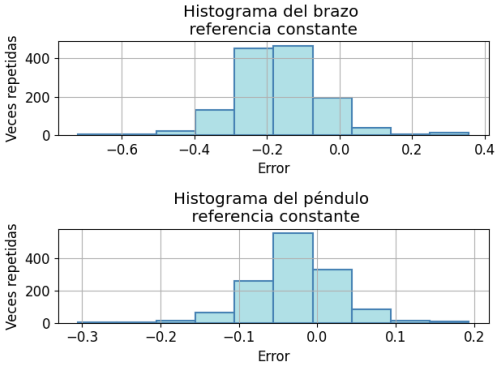

Figura 3.26: Histograma de referencia constante en la prueba práctica con la ecuación (2.6)

Se realiza una prueba con referencias constantes que cambiaban, y los resultados son mostrados en la Figura 3.27. El desempeño en cuanto al estado x_3 es aceptable, sigue la referencia con oscilaciones, mientras que en el estado x_1, es donde cambian las referencias,

sigue la referencia con oscilaciones y error en estado estable. En el histograma de la Figura 3.28 se observa un sesgo a la izquierda, en el estado x_1, debido al error en estado estable.

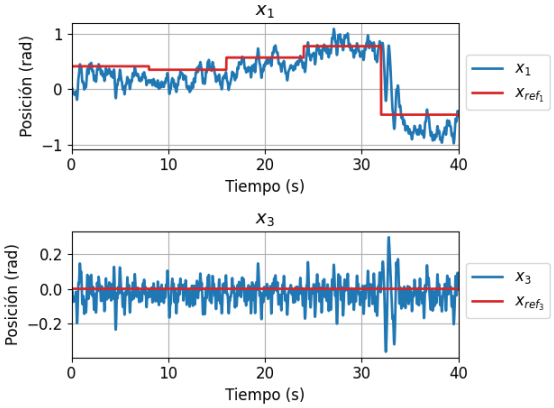

Figura 3.27: Múltiples constantes en la prueba práctica con la ecuación (2.6)

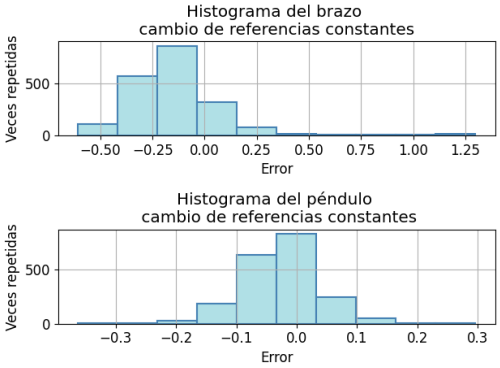

Figura 3.28: Histograma de múltiples constantes en la prueba práctica con la ecuación (2.6)

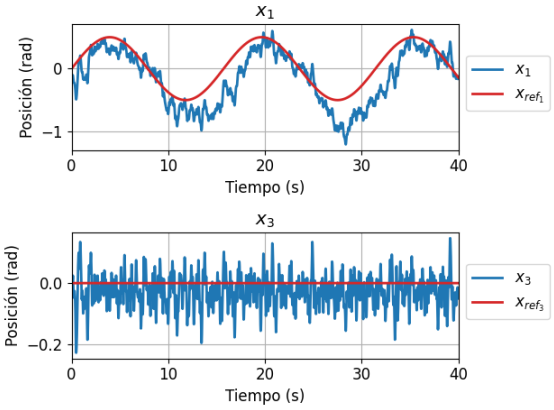

Figura 3.29: Referencia variante con el tiempo en la prueba práctica con la ecuación (2.6)

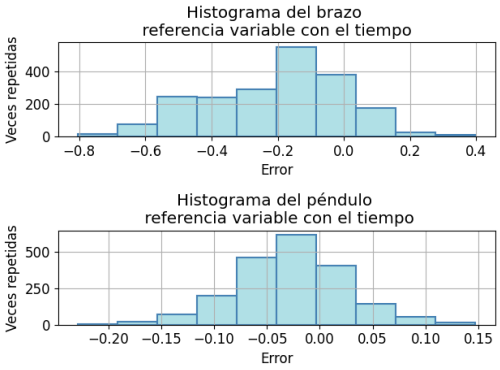

Figura 3.30: Histograma de referencia variante con el tiempo en la prueba práctica con la ecuación (2.6)

Se realiza una prueba con una señal de referencia variante con el tiempo $sin(0.2*t)$, (Figura 3.29) en la cual el estado x_3 se mantiene en la referencia constante, simultáneamente la referencia del estado x_1 es seguida por momentos, con oscilaciones. En el histograma de la Figura 3.30, donde se observa un sesgo hacia los números negativos, debido al error al estado estable.

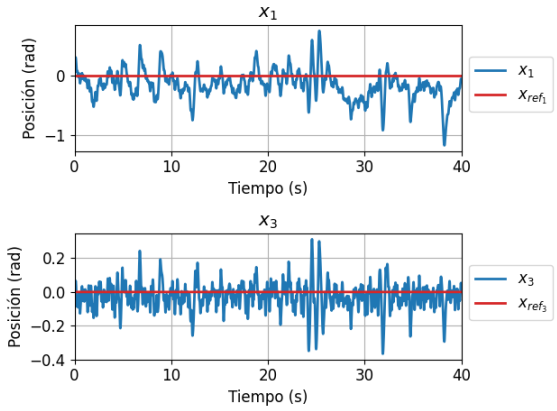

Figura 3.31: Resistencia a perturbaciones en la prueba práctica con la ecuación (2.6)

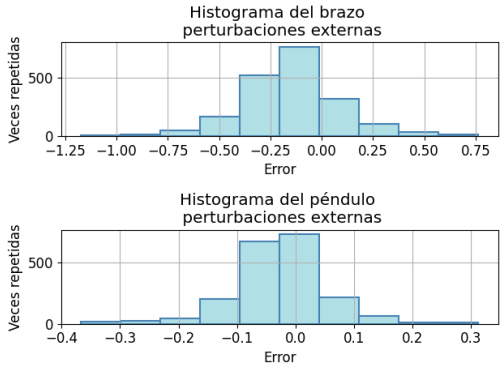

Figura 3.32: Histograma de resistencia a perturbaciones en la prueba práctica con la ecuación (2.6)

La última prueba que se realiza es la de perturbaciones externas, en esta prueba se golpea el péndulo para alterar el sistema, el controlador tiene como referencia una constante para las dos estados. Los resultados de esta prueba se muestran en la Figura 3.31, donde el estado x_3 no es afectado, en cambio el estado x_1 es afectado, al igual a las pruebas anteriores hay presencia de error en estado estable, esto se confirma con el histograma de la Figura 3.32, donde se observa un sesgo hacia los números negativos.

3.6. Conclusiones del Capítulo

En este capítulo se presentaron los resultados obtenidos de las distintas pruebas realizadas. Las simulaciones de prueba del sistema lineal, el modelo del motor de corriente directa, resultaron satisfactorios, aunque presenta un error en estado estable, pero es mínimo, exceptuando esto el controlador logró seguir las referencias en las tres pruebas realizadas y durante el entrenamiento se presenta una curva de aprendizaje aunque sin llegar a la máxima recompensa posible.

En cuanto al péndulo rotacional invertido, en simulación con el modelo no lineal,

las tres funciones de recompensa lograron seguir la referencia en el estado x_3, y solamente la función de recompensa de la ecuación 2.4 no logró seguir la referencia en el estado x_1, esto porque esta función no toma en cuenta ese estado durante el entrenamiento, por otro lado la ecuación que tiene el mejor desempeño basado en el análisis cuantitativo de las pruebas fue la ecuación 2.5. En cuanto a la prueba con el péndulo rotacional invertido en el laboratorio solamente el controlador entrenado con la función de recompensa propuesta logró tener un entrenamiento exitoso, demostrando tener aprendizaje inclusive en ambientes que favorecen poco debido a las condiciones iniciales o errores en los sensores.

Se observaron picos de magnitud que en los estados de la prueba física del péndulo rotacional invertido, Figuras 3.25, 3.27, 3.29 y 3.31, son debidos a que la red neuronal del actor toma la decisión de cambiar la acción de control inclusive ya estando el estado en la referencia, aunque como se mencionó estos cambios son solamente de un Δu, considerablemente pequeño, esto puede ser corregido o al menos disminuido utilizando el esquema de la Figura 2.5.b en vez del inciso a que fue el que se uso para esta prueba, no se realizo para este trabajo porque este aspecto se noto en la redacción de este trabajo. Otro resultado indeseable fue el error en estado estable, el cual esta presente en los controladores, la forma de corregir este error sería por medio del entrenamiento, prestando especial atención a las constantes de sesgo que son las que podrían ser las responsables de esto.

Se notó que cuanto a las condiciones iniciales para el control con la planta física del péndulo rotacional invertido, Figura 3.20, esta pudieron haber afectado el entrenamiento debido a que un entrenamiento más completo cuenta con la mayor cantidad de escenarios posibles, esto no se cumple, ya que en la Figura 3.20 se ve cómo se entrenó más las condiciones iniciales que se encuentran en los extremos de los rangos. Este problema tendría solución si se tuviera un aparato parecido a un soporte el cual permite colocar el péndulo en una posición deseada y una vez inicie el entrenamiento suelte el péndulo para no intervenir con este proceso, estantes mencionado es difícil de realizar con solamente el uso de la mano, la otra posible solución es corregir el sensor del péndulo para distinguir la orientación y empezar el entrenamiento con el péndulo boca abajo.

Para la prueba con la planta física del péndulo rotacional invertido se aumentaron las ganancias de la función de recompensa, esto para compensar que eran menos entrenamientos, en la simulación se entrenaban hasta $4,000$ épocas, pero esto es imposible en la prueba real, ya que tomaría un mayor tiempo por lo que los entrenamientos eran de aproximadamente 600 épocas. Al aumentar la magnitud de la recompensa se hace más rápido el aprendizaje, debido a que la retroalimentación es más grande teniendo un mayor efecto en los coeficientes de las redes neuronales, pero esto tiene un límite, si se llegase a poner una función de recompensa que regrese una magnitud excesiva dañará el entrenamiento, esto se observaría en la curva de aprendizaje con oscilaciones marcadas en la magnitud de la recompensa obtenida por época, no se encontró en el estado del arte un método para encontrar estos valores así que los presentados en este trabajo fueron obtenidos por medio de la experimentación.

Se observó que el error en el tiempo en la prueba física del péndulo rotacional invertido, Figura 3.21, afecta el cálculo de la velocidad lo que termina afectando tanto el entrenamiento como el desempeño del controlador de una manera considerable, ya que no está recibiendo información correcta a la red neuronal acerca del estado del péndulo rotacional invertido, este error no tiene ninguna otra solución más que repetir todo el experimento con la corrección en el programa en el cálculo del tiempo, lo cual no se pudo realizar, ya que este fue detectado al momento de redactar toda la información.

Capítulo 4

Conclusiones

4.1. Introducción

Se expuso todo lo relevante relacionado durante esta investigación en lo capítulos anteriores han logrado los objetivos planteados al inicio, aunque los resultados del desempeño del controlador presentan problemas como: oscilaciones y error en el estado estable, por lo que en este capítulo se aborda estos, posibles soluciones, así como las conclusiones generales de todo el proceso de aprendizaje de las redes neuronales, trabajos futuros que surgen a raíz de las cuestiones que surgieron a la par que se desarrolló este trabajo.

4.2. Conclusiones Generales

Las conclusiones obtenidas de la realización de esta tesis son:

- La implementación de un controlador por refuerzo basado en actor crítico puede ser utilizado para el control de sistemas lineales o no lineales, tanto en sistemas físicos, como simulados.

- Los controladores con actor discreto presentan oscilaciones en estado estable, debido a que sus variaciones en la señal de control son grandes para el sistema, estas oscilaciones son proporcionales a la sensibilidad del sistema y del delta de control Δu propuesto.

- Los resultados de esta tesis señalan que los controladores aquí expuestos son sensibles a perturbaciones externas.

- El entrenamiento físico no logra generar una curva de aprendizaje para el controlador, como sí lo hace en simulación, debido a dos factores los cuales cambian en comparación con la simulación, el primero es el hecho de que el tiempo de simulación o de muestreo en la simulación era de la mitad de magnitud, esto permite que controlador tome más decisiones en un lapso de tiempo menor, ayudando a la toma de decisiones, el segundo factor es el hecho de cuantos episodios se entrenaron, para la simulación se utilizaron $4,000$ contrario a los 600 de la prueba con la planta física, por eso las funciones de recompensa de la ecuación (2.4) y ecuación (2.5) no aprendieron a controlar el sistema, esto resalta que la función de recompensa propuesta es mejor cuando las condiciones del entrenamiento no son óptimas, debido a que hace una clara distinción entre las acciones que contribuyen a llegar a la referencia y a las que no.

- Un beneficio de la función de recompensa propuesta, ecuación (2.6), es que en la evolución de la recompensa llega a estabilizarse incluso en el entrenamiento con el sistema real, la cual presento múltiples fallas técnicas en su implementación, por lo que la función de recompensa propuesta funciona inclusive en circunstancias adversas.

- Si las condiciones del entrenamiento son las óptimas, la función de recompensa que mejor funciona es la ecuación (2.5), debido a que en las pruebas simuladas llega a tener menos oscilaciones en los estados al alcanzar la referencia.

- Las acciones discretas fueron propuestas a partir de la experiencia desarrollada durante la realización de las pruebas, no se encontró ningún otro trabajo que propusiera unas iguales, estas buscan imitar las acciones que llevaría a cabo un actor continuo, se podría considerar otro aporte este tipo de acciones discretas por parte de este trabajo.

- Es difícil encontrar todos los parámetros o características que debe tener el controlador para que funcione, por lo que este puede ser una gran desventaja al querer implementar este tipo de controladores, una solución sería encontrar un método para la proposición de estos basados en las propiedades del sistema que se desea controlar.

- Este tipo de controladores aún no son llevados a su máxima capacidad, solamente hay que observar la cantidad limitada de trabajo en este tema, pero si se logran desarrollar métodos para búsqueda de parámetros más efectivos, proposiciones de las arquitecturas de las redes neuronales y mejores funciones de recompensa, puede que estos métodos lleguen a rivalizar con los métodos basados en análisis matemáticos, que hoy en día llevan ventaja en el control de plantas complejas.

- La implementación física de este tipo de sistemas aún es difícil de realizar, debido a que se necesita de una gran cantidad de cálculos, y lo cual no es deseado para los controladores, ya que dependiendo de la planta pueden llegar a necesitar una reacción rápida para poder controlar al sistema.

- Se logró seguir referencias que no se usaron en el entrenamiento, en el caso del motor de CD, incluso se cambiaron los límites del voltaje, y aún así se logró tener un buen desempeño del controlador, esto contrasta con los resultados obtenidos en [Bejar and Moran, 2018], es debido a que en este método tanto el crítico como el actor se adapta al sistema, mientras que en el trabajo antes mencionado proponen una función ya establecida, esto hace que el resultado no capte bien las características del sistema.

- En la prueba física, debido a que el programa redondeaba una variable de tiempo, la velocidad presento cambios bruscos en su magnitud, lo que también influyó en un mal resultado en el funcionamiento del entrenamiento, sin embargo, la función propuesta en esta tesis si logró su objetivo, la razón de esto puede ser que es la única función que tomaba en cuenta todos los estados del sistema.

- Es posible que la resistencia tan débil hacia las perturbaciones se deba en gran medida al problema en el cálculo de la velocidad que se tuvo en la implementación en físico del controlador.

- El desempeño de este controlador, comparado con otros del estado del arte que suelen ser más utilizados, es comparable al uso de controlador proporcional integral derivativo, el error en estado estable es de la misma magnitud, pero el desempeño es inferior

a las técnicas de control como regulador lineal cuadrático o retroalimentación de estados, estos logran un error en estado estable mucho menor, con menos oscilaciones una vez alcanzada la referencia, esto mencionado es en comparación con el artículo de investigación [Akhtaruzzaman and Shafie, 2010].

4.3. Trabajos Futuros

1. Realizar más pruebas de la función de recompensa propuesta, para comprobar su funcionamiento, porque no se puede comprobar de manera teórica, debido a lo complejo que seria analizar cada uno de los posibles resultados con tantas variables que influyen en el proceso.

2. Buscar un procedimiento para encontrar los valores constantes de la función de recompensa propuesta.

3. Probar nuevas acciones en el experimento práctico realizado, con la intención de reducir las oscilaciones en estado estable y si es posible mejorar la resistencia a perturbaciones externas.

4. Realizar las mismas pruebas en cuanto a desempeño que se realizaron para este trabajo pero en vez de utilizar la red neuronal del actor con salidas a acciones discretas, donde cada una representa una acción, proponer una salida continua para control con lo cual la salida será directamente la acción de control del sistema.

5. Buscar, si es posible, un procedimiento para encontrar las condiciones iniciales de las redes neuronales del actor y del crítico, esto con el fin de mejorar el desempeño del entrenamiento.

6. Repetir las pruebas aquí presentadas, pero corrigiendo el error con la variable de tiempo necesario para el cálculo de las velocidades.

Apéndice A

Código del Algoritmo para el entrenamiento del control del motor de CD

```
# Probado en Python 3.8.10, usando Linux Mint 20.3

# Es neceseraio tener los siguientes paquetes

import sys
from scipy import integrate
import torch
import numpy as np
from torch._C import JITException
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.autograd import Variable
import matplotlib.pyplot as plt
import pandas as pd
from scipy.integrate import *
from datetime import date, datetime

# Parametros del modelo

# (J)     momento de inercia del rotor       0.01 kg.m^2
# (b)     friccion del motor      0.1 N.m.s
# (Ke)    constante de fuerza electromotriz  0.01 V/rad/sec
# (Kt)    constante del torque               0.01 N.m/Amp
# (R)     resitencia electrica               1 Ohm
# (L)     inductancia electrica              0.5 H


J = 3.72e-5
Ka = 0.0453
Kg = 0.0453
La = 3.4e-3
f = 5.23e-5
```

```
Ra = 2.30

# Parametros del aprendizaje

learning_rate = 300e-5
GAMMA = 0.99
num_steps = 20 # Segundos que dura el entrenamiento
max_episodes = 1500 # Número de episodios
epoch_print = 150 # Cada cuantas epocas imprime en terminal el resultado del entrenamiento
t_delta_control = 0.01 # Delta de accion de control
c1 = 20 # Constante funcion de recompensa
epsilon = 10 # Constante funcion de recompensa


# Parametros para control de la señal

MAX_VOLTAJE = 25
MIN_VOLTAJE = 0
DELTA_VOLTAJE = 0.01

# Modelo del sistema

def nolmodel(t,x,u):

xdot = np.array([0,0] , dtype=float)

xdot[0] = -(f/J)*x[0] + (Ka/J)*x[1]
xdot[1] = -(Kg/La)*x[0] - (Ra/La)*x[1] + (1/La)*u

return xdot


# Modelo de las redes neuronales, por cuestiones practicas ambas redes estan en la misma función

class ActorCritic(nn.Module):
def __init__(self, num_inputs, num_actions, hidden_size, learning_rate=3e-1):
super(ActorCritic, self).__init__()

self.num_actions = num_actions
self.critic_linear1 = nn.Linear(num_inputs, hidden_size)
self.critic_linear2 = nn.Linear(hidden_size, 1)

self.actor_linear1 = nn.Linear(num_inputs, hidden_size)
self.actor_linear2 = nn.Linear(hidden_size, num_actions)

def forward(self, state):
state = Variable(torch.from_numpy(state).float().unsqueeze(0))
value = F.hardtanh(self.critic_linear1(state))
value = self.critic_linear2(value)

policy_dist = F.hardtanh(self.actor_linear1(state))
policy_dist = F.softmax(self.actor_linear2(policy_dist), dim=1)

return value, policy_dist


# Parametros de las redes neuronales

num_inputs = 1
num_outputs = 3
hidden_size = 25

# Configuración del optimizador a usar para la retroalimentación
```

```
actor_critic = ActorCritic(num_inputs, num_outputs, hidden_size)
ac_optimizer = optim.Adam(actor_critic.parameters(), lr=learning_rate)

# Vectores donde se guarda la información del entrenamiento

all_lengths = []
average_lengths = []
all_rewards = []
entropy_term = 0

# Inicia el aprendizaje

for episode in range(max_episodes+1):

log_probs = []
values = []
rewards = []
steps = 0

# Condiciones inciales

t0 = 0
x0 = [0,0]#[(np.random.rand())*5,(np.random.rand())*0.5]
output_cntrl = 0
ref = [(np.random.rand())*500]
state = np.array( x0 + ref + [output_cntrl] + [0])
tf = num_steps
t_control = 0 # To manage the time of action's control

# Vector para guardar los estados

state_v = np.array([state])
t_v = np.array([t0])

# Bandera para terminar el entrenamiento

done = False

# Configuración de la simulación

x = RK45(lambda t,y: nolmodel(t,y,output_cntrl),t0,x0,tf,max_step=t_delta_control)

# Inicia una epoca

while(x.status == "running"):

# Es para llevar el muestreo de la simulación
t_control  = t_delta_control + t_control

# Se extreae el valor de la red neuronal del critico y la politica o acción del actor
value, policy_dist = actor_critic.forward( np.round( np.array( [state[2] - state[0] ] ) ,3) )
value = value.detach().numpy()[0,0]
dist = policy_dist.detach().numpy()
action = np.random.choice(num_outputs, p=np.squeeze(dist))
log_prob = torch.log(policy_dist.squeeze(0)[action])
entropy = -np.sum(np.mean(dist) * np.log(dist))

# Dependiendo de la accion se modifica la señal de control
if action == 0 and output_cntrl > MIN_VOLTAJE:
output_cntrl = output_cntrl - DELTA_VOLTAJE
if action == 1 and output_cntrl < MAX_VOLTAJE:
output_cntrl = output_cntrl + DELTA_VOLTAJE
```

```
if action == 2:
output_cntrl = output_cntrl

# Calculo necesesario para la función de recompensa
d_k = c1*np.abs(state[2] - x.y[0])

# Simula hasta que supere el tiempo de muestreo
while t_control > x.t :
x.step()

# Calculo de la función de recompensa
d_k_1 = c1*np.abs(state[2] - x.y[0])

if d_k_1 < epsilon:
reward = 100
else:
reward = d_k - d_k_1

# Se guarda la información del nuevo estado despues de la acción de control
new_state = np.concatenate([x.y, np.array( ref + [output_cntrl] + [reward])])

# Se guardan los valores de las redes neuronales y se calcula la entropia
# para calcular más adelante las funciones de perdida
rewards.append(reward)
values.append(value)
log_probs.append(log_prob)
entropy_term += entropy

# Se preparan las variables para el siguiente estado
steps = steps + 1
state = new_state
state_v = np.concatenate((state_v,np.array([state])),axis=0)
t_v = np.concatenate((t_v,np.array([x.t])),axis=0)


# Si se supero los limites del estado se pone como verdadera la bandera
# para terminar la simulación
if np.abs(x.t) > tf - t_delta_control:
done = True

# Cuando se termina la simulación o se superan los limites de los estados
if done or x.status == "finished":

# Se obtiene el ultimo valor del critico

Qval, _ = actor_critic.forward(np.round( np.array( [state[2] - state[0] ] ) ,3))

# Se preparna los datos para el calculo de la funcion de ventaja
Qval = Qval.detach().numpy()[0,0]
all_rewards.append(np.sum(rewards))
all_lengths.append(steps)
average_lengths.append(np.mean(all_lengths[-10:]))

# Imprime y grafica como va el entrenamiento para estar monitoreando
if episode % epoch_print == 0:
sys.stdout.write("episode: {}, reward: {}, total length: {}, average length: {} \n".format(episode, np.sum(rewards), steps, average

np.savetxt(name, state_v, delimiter=',')
name = str(datetime.now().strftime("%H-%M-%S-%b-%d")) + "-" + str(learning_rate) + "-" + str(max_episodes) + "-" + str(episode) + "
torch.save( actor_critic,name)
name = str(datetime.now().strftime("%H-%M-%S-%b-%d")) + "-" + str(learning_rate) + "-" + str(max_episodes) + "-" + str(episode) + "
np.savetxt(name, np.array(all_rewards), delimiter=',')
```

```
plt.figure(str(episode))
plt.subplot(411)
plt.plot(t_v,state_v[:,[-1]])
plt.subplot(412)
plt.plot(t_v,state_v[:,[0,2]])
plt.subplot(413)
plt.plot(t_v,state_v[:,[0,1]])
plt.subplot(414)
plt.plot(t_v,state_v[:,[-2]])

break

# Calculo de los valores de r + V(sk+1) con los vectores despues
# de cada epoca
Qvals = np.zeros_like(values)
for t in reversed(range(len(rewards))):
Qval = rewards[t] + GAMMA * Qval
Qvals[t] = Qval

# Se actualizan y se dan formato de tensor a los datos
values = torch.FloatTensor(values)
Qvals = torch.FloatTensor(Qvals)
log_probs = torch.stack(log_probs)

# Se calcula la funcion de ventaja
advantage = Qvals - values

# Se calculan las funciones de perdida para cada red neuronal
actor_loss = (-log_probs * advantage).mean()
critic_loss = 0.5 * advantage.pow(2).mean()
ac_loss = actor_loss + critic_loss + 0.001 * entropy_term

# Se hace la propagración hacia atras
ac_optimizer.zero_grad()
ac_loss.backward()
ac_optimizer.step()



# Graficacion y se guardado en archivos de los resultados

smoothed_rewards = pd.Series.rolling(pd.Series(all_rewards), 10).mean()
smoothed_rewards = [elem for elem in smoothed_rewards]

name = str(datetime.now().strftime("%H-%M-%S-%b-%d")) + "-" + str(learning_rate) + "-" + str(max_episodes) + "-rewar
np.savetxt(name, all_rewards, delimiter=',')
name = str(datetime.now().strftime("%H-%M-%S-%b-%d")) + "-" + str(learning_rate) + "-" + str(max_episodes) + "-smoot
np.savetxt(name, smoothed_rewards, delimiter=',')

plt.figure()
plt.plot(all_rewards)
plt.plot(smoothed_rewards)
plt.plot()
plt.xlabel('Episode')
plt.ylabel('Reward')
plt.show()
```

___

Referencias

[Akhtaruzzaman and Shafie, 2010] Akhtaruzzaman, M. and Shafie, A. A. (2010). Modeling and control of a rotary inverted pendulum using various methods, comparative assessment and result analysis. In *2010 IEEE International Conference on Mechatronics and Automation*, pages 1342–1347. IEEE.

[Arulkumaran et al., 2019] Arulkumaran, K., Cully, A., and Togelius, J. (2019). Alphastar: An evolutionary computation perspective. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 314–315.

[Basilio and Moreira, 2004] Basilio, J. C. and Moreira, M. V. (2004). State-space parameter identification in a second control laboratory. *IEEE Transactions on Education*, 47(2):204–210.

[Bejar and Moran, 2018] Bejar, E. and Moran, A. (2018). Control of a two-dimensional magnetic positioning system with deep reinforcement learning and feedback linearization. In *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 909–912. IEEE.

[Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[Diao, 2016] Diao, X. (2016). Modular control of a rotary inverted pendulum system. *Purdue University*, page 3.

[François-Lavet et al., 2018] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G.,

and Pineau, J. (2018). An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*.

[Grondman et al., 2011] Grondman, I., Vaandrager, M., Busoniu, L., Babuska, R., and Schuitema, E. (2011). Efficient model learning methods for actor–critic control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):591–602.

[Laura Graesser, 2019] Laura Graesser, W. L. K. (2019). *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. Pearson.

[MacCrimmon, 1968] MacCrimmon, K. R. (1968). Descriptive and normative implications of the decision-theory postulates. In *Risk and uncertainty*, pages 3–32. Springer.

[Ormrod, 2011] Ormrod, J. E. (2011). *Human learning*. Pearson Higher Ed.

[Peterson, 2017] Peterson, M. (2017). *An introduction to decision theory*. Cambridge University Press.

[Si and Wang, 2001] Si, J. and Wang, Y.-T. (2001). Online learning control by association and reinforcement. *IEEE Transactions on Neural networks*, 12(2):264–276.

[Silver et al., 2018] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

[Spielberg et al., 2017] Spielberg, S., Gopaluni, R., and Loewen, P. (2017). Deep reinforcement learning approaches for process control. In *2017 6th international symposium on advanced control of industrial processes (AdCONIP)*, pages 201–206. IEEE.

[Sun et al., 2019a] Sun, Q., Du, C., Duan, Y., Ren, H., and Li, H. (2019a). Design and application of adaptive pid controller based on asynchronous advantage actor–critic learning method. *Wireless Networks*, pages 1–11.

[Sun et al., 2019b] Sun, Q., Ren, H., Duan, Y., and Yan, Y. (2019b). The adaptive pid

controlling algorithm using asynchronous advantage actor-critic learning method. In *International Conference on Simulation Tools and Techniques*, pages 498–507. Springer.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Vallejo, 2021] Vallejo, M. A. P. (2021). MiguelAngelPimentelVallejo/rewards_comparison_A2C: The repository is empty but not for much time.

[Wu and Tian, 2016] Wu, Y. and Tian, Y. (2016). Training agent for first-person shooter game with actor-critic curriculum learning.

[Xiong et al., 2019] Xiong, Y., Zheng, G., Xu, K., and Li, Z. (2019). Learning traffic signal control from demonstrations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2289–2292.

[Yu, 2018] Yu, W. (2018). Controller design for mechatronic rotary inverted pendulum (part 1 and part 2). In *2018 ASEE Mid-Atlantic Section Spring Conference*.

[Zheng et al., 2020] Zheng, Y., Li, X., and Xu, L. (2020). Balance control for the first-order inverted pendulum based on the advantage actor-critic algorithm. *International Journal of Control, Automation and Systems*, 18(12):3093–3100.