

Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Programa de Licenciatura en Ingeniería Electrónica

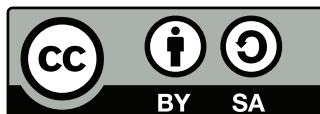


**Método basado en aprendizaje reforzado
para el control automático de una
planta no lineal**

Informe de Trabajo Final de Graduación para optar por el título de
Ingeniero en Electrónica con el grado académico de Licenciatura

Oscar Andrés Rojas Fonseca

Cartago, 11 de junio, 2024



Este trabajo titulado *Método basado en aprendizaje reforzado para el control automático de una planta no lineal* por Oscar Andrés Rojas Fonseca, se encuentra bajo la Licencia Creative Commons **Atribución-ShareAlike 4.0 International**.

Para ver una copia de esta Licencia, visite <http://creativecommons.org/licenses/by-sa/4.0/>.

Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Oscar Andrés Rojas Fonseca

Cartago, 17 de junio de 2024

Céd: 1-1696-0962

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Trabajo Final de Graduación
Acta de Aprobación

Defensa de Trabajo Final de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura

El Tribunal Evaluador aprueba la defensa del trabajo final de graduación denominado *Método basado en aprendizaje reforzado para el control automático de una planta no lineal*, realizado por el señor Oscar Andrés Rojas Fonseca y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

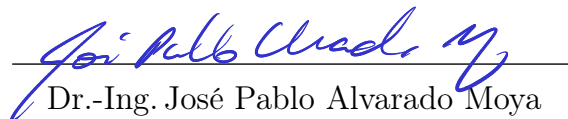
Miembros del Tribunal Evaluador



Dr. Alfonso Chacón Rodríguez
Profesor Lector



M.Sc. Javier Mauricio Pérez Rodríguez
Profesor Lector



Dr.-Ing. José Pablo Alvarado Moya
Profesor Asesor

Cartago, 11 de junio, 2024

Resumen

La aplicación del aprendizaje reforzado en áreas de control automático representa un paso en la dirección de optimización de tareas que requieren capacidades de autonomía y flexibilidad, rubros que el RL ofrece con métodos como el DQN y el PPO, en su implementación con redes neuronales artificiales. El presente proyecto explora estos métodos en una planta de péndulo amortiguado de motor con hélice, con una estrategia que recompensa el bajo error angular y baja velocidad terminal. Se emplearon entornos virtuales para las pruebas de entrenamiento de control. Se evidencia la necesidad de ajustes en la función de recompensa para dar con el punto de control deseado y acercarse a cumplir con criterios usuales de control.

Palabras clave: Aprendizaje reforzado, DQN, PPO, función de recompensa, control automático

Abstract

The application of reinforced learning in areas of automatic control represents a step in the direction of optimization of tasks that require autonomy and flexibility capabilities, items that RL offers with methods such as DQN and PPO, in its implementation with artificial neural networks. The present project explores these methods in a damped pendulum propeller motor plant, with a strategy that rewards low angular error and low terminal velocity. Virtual environments were used for control training tests. The need for adjustments to the reward function to find the desired control point and come close to meeting the usual control criteria is evident.

Keywords: Reinforcement learning, DQN, PPO, reward function, automatic control

a mis queridos abuelos

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin el apoyo de mis padres, Oscar Rojas Arce y Miriam Fonseca Elizondo, que día a día trabajaron para darme el estudio y los valores que tanto los caracterizan y me formaron como persona responsable y agradecida.

Agradecer a mi pareja Heikel Oporto Montero, por su paciencia y apoyo constante durante todo mi estudio universitario, que sin su presencia todo sería muy difícil de sobrellevar.

Agradezco a mis abuelos, Jose Miguel Fonseca Fonseca y Cristina Elizondo Pérez, que ya no se encuentran conmigo pero los recordaré con mucho cariño por su presencia durante toda mi infancia y valores sobresalientes.

Agradecer a los profesores de la Escuela de Ingeniería en Electrónica por su trabajo duro y apoyo en todos los trabajos realizados, especialmente al profesor José Pablo Alvarado Moya por su paciencia, sabiduría y calidad de docente.

Oscar Andrés Rojas Fonseca

Cartago, 17 de junio de 2024

Índice general

Índice de figuras	III
Índice de tablas	V
Lista de símbolos y abreviaciones	VI
1. Introducción	1
1.1. Aprendizaje automático en control	1
1.2. Control con aprendizaje reforzado	2
1.3. Modelo controlador de planta de laboratorio PAMH con RL	3
1.4. Objetivos y estructura del documento	4
2. Marco teórico	5
2.1. Péndulo amortiguado de motor con hélice	5
2.1.1. Modelo analítico del PAMH	6
2.2. Redes neuronales artificiales (RNA)	7
2.3. Aprendizaje reforzado	8
2.3.1. Conceptos base del RL	9
2.3.2. Deep Q-Network (DQN)	12
2.3.3. Optimización de política próxima (PPO)	13
2.4. Métricas de evaluación	14
2.4.1. Función de recompensa	14
2.4.2. Pérdida del modelo	14
2.4.3. Error de posición angular	15
3. Control de PAMH basado en aprendizaje reforzado	16
3.1. Selección de modelos de RL	16
3.2. Preparación de entornos virtuales	17
3.3. Parametrización de los modelos	19
3.3.1. Función de recompensa	19
3.3.2. Estructura de las RNA en DQN y PPO	21
3.3.3. DQN	22
3.3.4. PPO	23
3.3.5. Procesamiento y recopilación de datos	25
3.4. Evaluación de los modelos	25

4. Resultados y análisis	26
4.1. Implementación de métodos RL	26
4.1.1. Preparación del entorno virtual	26
4.1.2. Modelo DQN	28
4.1.3. Modelo PPO	28
4.2. Evaluación del desempeño de los métodos RL	30
5. Conclusiones y recomendaciones	33
5.1. Conclusiones	33
5.2. Recomendaciones	34
Bibliografía	35
A. Algoritmo del método DQN	37
B. Algoritmo del método PPO	38

Índice de figuras

1.1.	Proyección de la evolución de los sistemas de control. Fuente: [2].	2
1.2.	Diagrama de etapas involucradas en la solución. En gris se muestran las etapas realizadas en [6]. En verde se muestran las etapas correspondientes a este trabajo y en blanco las etapas a realizar en un trabajo futuro. Adaptado de: [6].	3
1.3.	Diagrama de control para la PAHM. La línea punteada de color rojo muestra el bloque a desarrollar. Fuente: [6].	3
1.4.	Etapas del entrenamiento del modelo de RL usando la red neuronal imitadora (RNAM) para simular el PAMH real. La línea punteada de color rojo muestra el bloque de implementación y entrenamiento. En gris el producto del trabajo en [6].	4
2.1.	Modelo simplificado del PAMH. Fuente: [8]	5
2.2.	Modelo de neurona artificial. Fuente: [11].	7
2.3.	Modelo red neuronal artificial. Fuente: [12].	8
2.4.	Etapas básicas del RL. Fuente: [12].	9
2.5.	Efecto de la selección de la probabilidad ε en el desempeño del agente. Fuente [13].	12
2.6.	Resumen de categorización del RL [12].	12
3.1.	Función de recompensa con $\theta_{objetivo} = 45^\circ$	21
3.2.	$\theta_{correcto}$ con $\theta_{objetivo} = 45^\circ$	21
3.3.	Estructura de la RNA utilizada en el método DQN.	23
3.4.	Estructura de la RNA utilizada en el método PPO.	24
4.1.	Duración del episodio en prueba de implementación DQN con entorno <i>Cart-Pole</i>	27
4.2.	Proceso de entrenamiento con la implementación PPO con entorno <i>Pendulum</i>	27
4.3.	Proceso de entrenamiento recompensa-pérdida de la implementación DQN con entorno PAMH.	29
4.4.	Últimos ángulos muestreados de los episodios del entrenamiento DQN con PAMH. La línea tenue en el fondo representa la señal completa, y la línea oscura dicha señal filtrada para rescatar su tendencia.	30

-
- 4.5. Recompensa promedio de los entrenamientos con diferentes valores de varianza σ^2 . La curva rosa contempla $\sigma^2 = 0,001$, la curva verde contempla $\sigma^2 = 0,01$ y la curva morada contempla el barrido en el rango de $\sigma^2 = [0,01, 0,001]$. Las líneas tenues en el fondo representan la señal completa, y las líneas oscuras dichas señales filtradas para rescatar sus tendencias. 30
- 4.6. Últimos ángulos muestreados de los episodios en el entrenamiento. La curva rosa contempla $\sigma^2 = 0,001$, la curva verde contempla $\sigma^2 = 0,01$ y la curva morada contempla el barrido en el rango de $\sigma^2 = [0,01, 0,001]$. Las líneas tenues en el fondo representan la señal completa, y las líneas oscuras dichas señales filtradas para rescatar sus tendencias. 31
- 4.7. Curva de respuesta del modelo DQN ángulo respecto al tiempo. 31
- 4.8. Curva de respuesta del modelo PPO ángulo respecto al tiempo. 32

Índice de tablas

3.1. Matriz de Pugh de los métodos de RL candidatos.	16
3.2. Equivalencia de valores continuos y discretos seleccionados para el método DQN.	22

Lista de símbolos y abreviaciones

Abreviaturas y Siglas

DQL	<i>Deep Q-Learning</i>
DQN	<i>Deep Q-Network</i>
IA	Inteligencia Artificial
ITCR	Instituto Tecnológico de Costa Rica
MDP	Procesos de decisión de Markov
PAMH	Péndulo Amortiguado de Motor con Hélice
PPO	Optimización de Política Próxima
PWM	Modulación por Ancho de Pulso
ReLU	Unidad lineal rectificada
RL	Aprendizaje Reforzado
RNA	Red Neuronal Artificial
RNAM	Red Neuronal Artificial mimetizadora
SIP-Lab	Laboratorio de Procesamiento de Imágenes y Señales

Notación general

\mathbf{A}	Matriz.
$\mathbf{A} =$	$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$
\mathbf{x}	Vector.
$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]^T =$	$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$
y	Escalar.

Capítulo 1

Introducción

1.1. Aprendizaje automático en control

Los sistemas de control automático en la ingeniería buscan satisfacer criterios de optimalidad preestablecidos. Algunos ejemplos de aplicación de técnicas de control son los movimientos precisos de los brazos robóticos en una planta de ensamblaje, la autonomía de procesos en los vehículos automotores y eléctricos, el funcionamiento de los reactores químicos o nucleares, el funcionamiento de equipos médicos, entre otros, todos buscando garantizar comportamientos que optimicen el rendimiento o costo de recursos [1].

Con el paso de los años el control automático ha reaccionado a cambios y demandas que la industria exige. Se busca optimizar funciones de manera que se alcancen capacidades de autonomía y flexibilidad, mientras se asegura la precisión y eficiencia de los sistemas. Sin embargo, las técnicas utilizadas en control en general logran solo una de estas capacidades. Por ejemplo, los sistemas de control programables ofrecen flexibilidad en el sistema, pero si se cambia ligeramente la configuración de la planta, el control deja de funcionar, evidenciando su falta de autonomía; y por otro lado, las automatizaciones permiten la autonomía de los equipos, pero únicamente responden a un tipo de maquinaria, por lo que no ofrecen flexibilidad [2]. La figura 1.1 ilustra que las nuevas tecnologías en robótica e inteligencia artificial IA buscan ofrecer las dos capacidades mencionadas y de manera consecuente, llevan a un salto de calidad, seguridad y rentabilidad [2] [3].

Los métodos clásicos de diseño en control, se basan a menudo en modelos matemáticos lineales, que no aplican a sistemas que presentan dinámicas no lineales o perturbaciones externas significativas. Dentro de los beneficios de utilizar IA para el control de equipos o sistemas se tiene, primero, la adaptabilidad en entornos inciertos. Las plantas a controlar, incluyendo robots, deberán operar en ambientes con obstáculos o cambios imprevisibles. Los métodos de control tradicionales se basan en respuestas preprogramadas, que pueden no funcionar en situaciones imprevistas. El aprendizaje reforzado (RL, por sus siglas en inglés) permite a “agentes” aprender por ensayo y error, adaptando su comportamiento en función de recompensas recibidas por acciones específicas. Esto los hace menos susceptibles

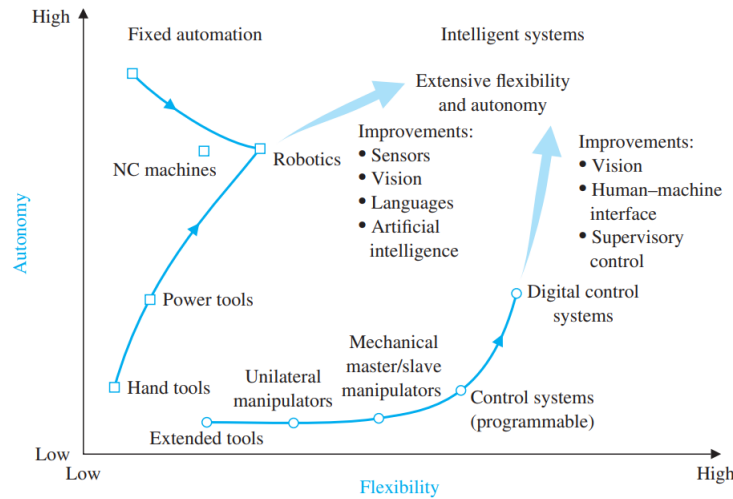


Figura 1.1: Proyección de la evolución de los sistemas de control. Fuente: [2].

a fallar a la hora de manejar lo inesperado, e incluso habilita a los sistemas a continuar aprendiendo durante su operación; pueden optimizar su comportamiento aun si su entorno cambia [4] [5].

Tareas usualmente invariantes junto con una programación estática, dificultan al controlador el adaptarse a situaciones imprevistas o descomponer tareas complejas en una serie de acciones más sencillas, resultando en todo un reto para el desarrollo con métodos de control tradicionales. El RL destaca en este tipo de situaciones. Aprender de las recompensas asociadas a cada acción permite que el sistema pueda desarrollar una estrategia o política para realizar trabajos de especial cuidado, incluso sin programación explícita para cada paso [4].

Al considerar sistemas no lineales de alta complejidad, el desarrollo de sistemas de control clásico moderno requieren conocimientos especializados, punto en el que las nuevas prácticas del RL alivianan esta carga. Al proporcionar una estructura general para el aprendizaje, el RL permite a los sistemas adquirir las habilidades necesarias a través de la interacción con el entorno [5].

1.2. Control con aprendizaje reforzado

En el Laboratorio de Procesamiento de Imágenes y Señales (SIP-Lab) de la Escuela de Ingeniería Electrónica del ITCR se experimenta con la aplicación de RL en el control para su posterior incorporación en actividades de investigación y docencia.

En la figura 1.2, se ilustran las fases del marco experimental a utilizar. Como planta física se emplea un péndulo amortiguado con motor y hélice (PAMH). Como primer paso, se evaluó una red neuronal imitadora de la planta física como parte del proyecto de graduación de Brenes Alfaro [6], que resultó en un modelo con características similares a la planta real. Este modelo permite experimentar sin riesgo de dañar la planta física real

durante los procesos de aprendizaje

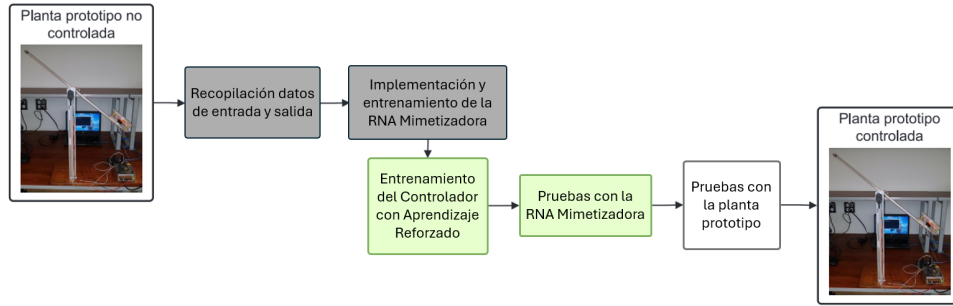


Figura 1.2: Diagrama de etapas involucradas en la solución. En gris se muestran las etapas realizadas en [6]. En verde se muestran las etapas correspondientes a este trabajo y en blanco las etapas a realizar en un trabajo futuro. Adaptado de: [6].

Ya que se cuenta con un modelo imitador, el siguiente paso en el desarrollo del proyecto correspondiente al entrenamiento de un modelo controlador de la planta de laboratorio. En la figura 1.3 se muestra la estructura general del sistema controlador, resaltada la sección que corresponde al presente proyecto, que se utilizará con los interruptores A y C abiertos y B cerrado. En iteraciones posteriores corresponderá desconectar B y conectar la planta real con A.

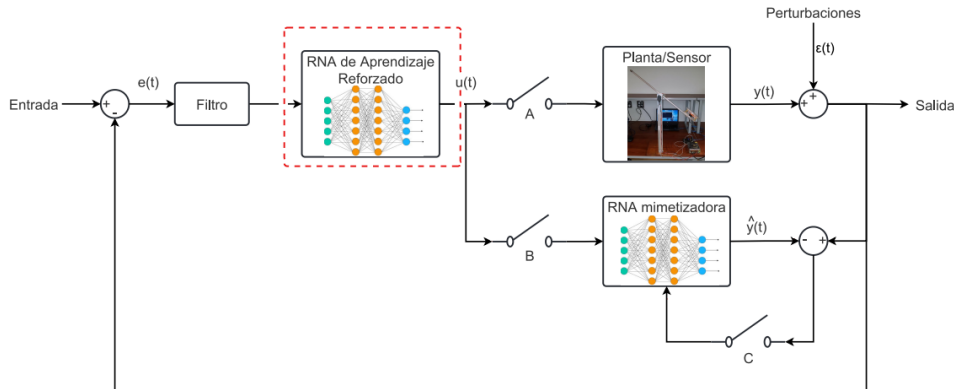


Figura 1.3: Diagrama de control para la PAMH. La línea punteada de color rojo muestra el bloque a desarrollar. Fuente: [6].

1.3. Modelo controlador de planta de laboratorio PAMH con RL

Este trabajo corresponde entonces a una segunda etapa de un proyecto de control del PAMH mediante técnicas de RL y redes neuronales artificiales. En la figura 1.4 se muestra la distribución de las etapas del presente trabajo. El primer paso consiste en recopilar, estudiar y calificar los modelos de RL para su selección como método a implementar. Seguidamente se entrena el modelo incorporando la implementación de un modelo RNAM

virtual de la planta, con ajustes a sus hiperparámetros. Por último, se evalúa el desempeño en el control del entorno PAMH.

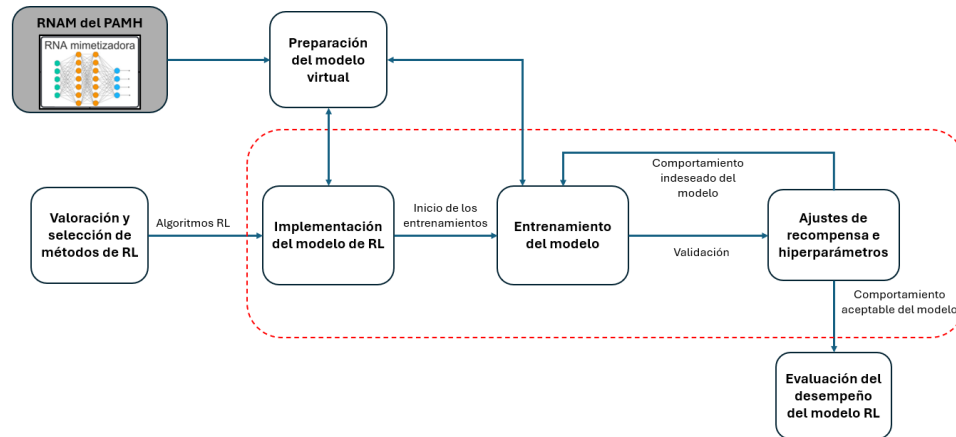


Figura 1.4: Etapas del entrenamiento del modelo de RL usando la red neuronal imitadora (RNAS) para simular el PAMH real. La línea punteada de color rojo muestra el bloque de implementación y entrenamiento. En gris el producto del trabajo en [6].

1.4. Objetivos y estructura del documento

El objetivo general del presente proyecto es diseñar un sistema de aprendizaje automático para el control del ángulo de una planta no lineal PAMH. El primer objetivo específico se enfoca en la selección de un método de aprendizaje reforzado apto para el control no lineal. Esto último es necesario por las características del sistema PAMH. En la selección se deben tomar en cuenta los recursos y los algoritmos de RL disponibles; por lo tanto, es necesario calificar los métodos planteados en primera mano y su congruencia respecto al objetivo a cumplir.

Seguidamente, el segundo objetivo específico es implementar el método de RL seleccionado, utilizando el modelo mimetizador (RNAS) como planta a controlar. Es necesario hacer los ajustes respectivos al algoritmo y especificar la función de recompensa del método para lograr controlar el ángulo del péndulo.

Como tercer objetivo se deben evaluar los resultados del modelo de RL para el control de la RNAS. Se deben identificar las deficiencias y ventajas del caso.

El presente documento contiene en su capítulo 2 los principios teóricos del funcionamiento del aprendizaje reforzado, así como una explicación de los métodos utilizados. En el capítulo 3 se describe la secuencia de tareas realizadas para alcanzar los objetivos específicos mencionados anteriormente.

Por último, el capítulo 4 y el capítulo 5 muestran los resultados de los procesos de entrenamiento con las valoraciones de estos y se plantean conclusiones y recomendaciones.

Capítulo 2

Marco teórico

En el presente capítulo se repasan los conceptos que fundamentan el proceso de diseño de un controlador de una planta de laboratorio como el péndulo amortiguado a hélice, mediante prácticas de aprendizaje reforzado.

2.1. Péndulo amortiguado de motor con hélice

El péndulo amortiguado de motor con hélice (PAMH) se trata de un sistema extrapolado de un péndulo simple, compuesto de un motor con hélice, controlado por una señal de modulación por ancho de pulsos (*pulse-width modulation*, PWM), una masa pequeña colocada en contrapeso al motor, el péndulo resultante de los brazos de aluminio y pesos correspondientes, así como soportes de aluminio de baja fricción, componentes responsables de mantener la estructura. Un modelo simplificado del sistema se muestra en la figura 2.1 junto con las magnitudes y vectores utilizados en la generación de modelo analítico del sistema [7].

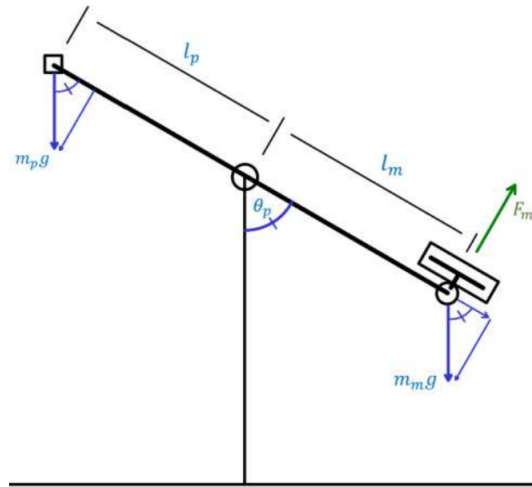


Figura 2.1: Modelo simplificado del PAMH. Fuente: [8]

El objetivo principal de dicho sistema es controlar la magnitud del ángulo θ_p medido entre la estructura y el péndulo, únicamente aplicando la señal PWM al motor que se encuentra a una distancia l_m de la estructura y ejerce una fuerza F_m , que provoca el movimiento de su masa m_m mientras, a una distancia de l_p del centro, se encuentra una masa m_p que contrarresta el movimiento.

2.1.1. Modelo analítico del PAMH

El modelado de sistemas físicos permite la elaboración de planes de trabajo con una base en común y bien sustentada en las características del entorno utilizado, permitiendo el diseño de controladores funcionales y eficientes. Este enfoque facilita la predicción del comportamiento del sistema bajo diversas condiciones operativas y la implementación de estrategias de control robustas, de manera que es común la elaboración de diagramas, como el caso de la figura 2.1 para la interpretación matemática [9].

El problema del péndulo suele ser abordado como un problema de leyes de Newton, donde se parte de la *ley de movimiento rotacional de Newton* aplicada al modelo simplificado de la figura 2.1.

$$\sum \tau = J_p \vec{a} \quad (2.1)$$

La sumatoria de torques τ corresponde a la inercia del cuerpo péndulo J_p por su aceleración angular \vec{a} [10], de manera que la ecuación (2.1) equivale a:

$$\begin{aligned} \tau_m - \tau_{mg} - \tau_\beta + \tau_{pg} &= J_p \vec{a} \\ l_m F_m - l_m m_m g \sin(\theta_p) - l_m \beta \dot{\theta}_p + l_p m_p g \sin(\theta_p) &= J_p \ddot{\theta}_p \end{aligned} \quad (2.2)$$

en donde β corresponde a la constante de rozamiento en el eje central de la planta, constante comúnmente despreciable. Además, para deflexiones pequeñas se linealiza la ecuación con la aproximación $\sin(\theta_p) \approx \theta_p$. Las ecuaciones de estado son:

$$\begin{aligned} x_1 &= \theta_p & x_2 &= \dot{\theta}_p & y &= x_1 = \theta_p \\ \begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{l_m \beta}{J_p} x_2 + (m_p l_p - m_m l_m) \frac{g}{J_p} x_1 + \frac{l_m}{J_p} F_m \end{cases} \end{aligned} \quad (2.3)$$

y la función de transferencia al aplicar el análisis en frecuencia:

$$T(s) = \frac{\theta_p(s)}{F_m(s)} = \frac{\frac{l_m}{J_p}}{s^2 + \frac{l_m \beta}{J_p} s + \frac{(l_m m_m - l_p m_p) g}{J_p}} \quad (2.4)$$

punto desde el que se puede empezar a plantear alguna estrategia de control clásico al sistema o entorno, pero limitando al rango angular propiciado por la aproximación del componente trigonométrico.

Una consideración del procedimiento analítico realizado es que solo aproxima a la planta real, pues el modelo de la ecuación (2.3) no considera el movimiento mecánico producto de la holgura en el eje, o factores de resistencia mecánica por los cables eléctricos que toman la señal angular o llevan la energía al motor, o alteraciones con histéresis producto del movimiento de esos cables, factores que, acumulados, acarrearán retos adicionales en el diseño de un controlador.

2.2. Redes neuronales artificiales (RNA)

Las RNA nacen de la experimentación para emular computacionalmente la capacidad del cerebro humano. El modelo usual de una neurona artificial se muestra en la figura 2.2 [11]. Los valores de entrada x_i se escalan por los pesos correspondientes w_{ni} , para luego sumarlos y transformarlos con la función de activación $\varphi(\cdot)$ (o también $f(\cdot)$), resultando en la salida [11]:

$$y_i = \varphi \left(\sum_{j=1}^m w_{ij} x_j \right) \quad (2.5)$$

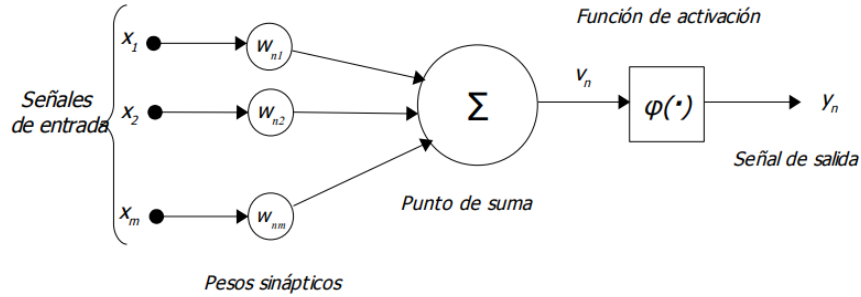


Figura 2.2: Modelo de neurona artificial. Fuente: [11].

Un número N de neuronas se interconecta dentro del sistema completo de la red neuronal artificial como se muestra en la figura 2.3. Esta red recibe los vectores de entrada \mathbf{x} y produce los vectores de salida \mathbf{y} . En el proceso, las capas ocultas producen los vectores $\mathbf{x}^{(n)}$ utilizando las matrices de pesos \mathbf{A}_i que guardan los pesos correspondientes a cada neurona [11] [12].

Por lo tanto, la salida de una capa es entonces:

$$\mathbf{x}^{(n+1)} = f(\mathbf{A}, \mathbf{x}^{(n)}) \quad (2.6)$$

La red neuronal completa se modela con la composición de M capas de la forma [12]:

$$\mathbf{y} = f_M(\mathbf{A}_M, \dots, f_2(\mathbf{A}_2, f_1(\mathbf{A}_1, \mathbf{x}))) \quad (2.7)$$

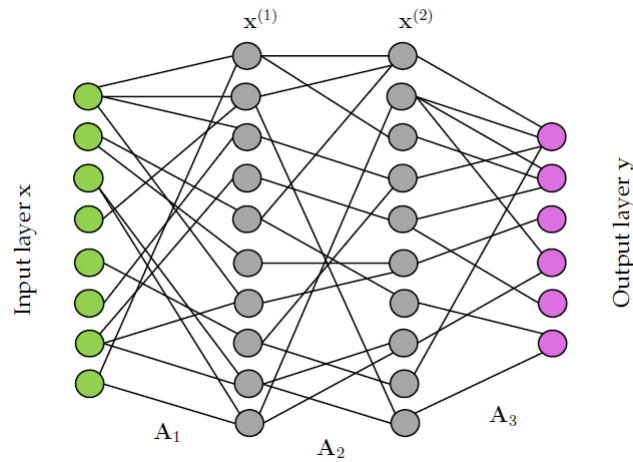


Figura 2.3: Modelo red neuronal artificial. Fuente: [12].

La función de activación más utilizada en la actualidad, por el bajo costo computacional que representa, sumado a características teóricas que propician el funcionamiento correcto de los procesos de aprendizaje, es la llamada función ReLU [12].

$$f_{ReLU}(x) = \begin{cases} 0 & \text{para } x \leq 0 \\ x & \text{para } x > 0 \end{cases} \quad (2.8)$$

En síntesis, estos modelos neuronales tienen capacidad de aproximar cualquier función, ajustando el número de capas, el número de neuronas, y las funciones de activación. El objetivo con estos modelos es, dado un conjunto de entrenamiento con datos de entrada \mathbf{x} para los que se conoce la salida \mathbf{y} , y dado un proceso de optimización, ajustar los pesos de cada capa, para que la diferencia entre las salidas que predice el modelo y las salidas \mathbf{y} del conjunto de entrenamiento se minimice poco a poco, hasta que el sistema “aprenda” a reproducir esas predicciones, generalizando lo aprendido incluso a patrones de entrada no vistos durante el proceso de entrenamiento [12].

2.3. Aprendizaje reforzado

En el aprendizaje reforzado (RL por sus siglas en inglés) se parte del supuesto que un agente debe aprender a interactuar en un entorno, de modo que a través de las acciones que el agente ejecuta en ese entorno, se maximice alguna función de recompensa. El agente tendrá algún estado que lo caracteriza (como su posición, su velocidad, etc.) y podrá realizar observaciones sobre el entorno, para decidir qué acciones tomar. Esto se ejemplifica en la figura 2.4 [12].

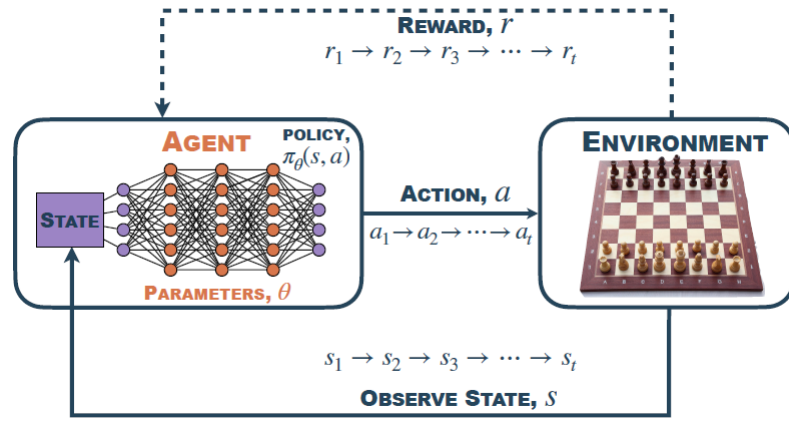


Figura 2.4: Etapas básicas del RL. Fuente: [12].

2.3.1. Conceptos base del RL

Política

En aprendizaje reforzado se conoce como política (*policy*) a la estrategia que utiliza un agente para decidir qué acción a tomar, dado su estado actual observable s . Existen varias formas de modelar las políticas. Por un lado, en los métodos de aprendizaje reforzado clásicos, la política es una función π determinista que mapea de estados a acciones, mientras que métodos más recientes tienden a expresar la política como [12]:

$$\pi(s, a) = Pr(a = a | s = s) \quad (2.9)$$

donde la política $\pi(s, a)$ es una función que expresa la probabilidad de tomar una acción a dado un estado s .

Recompensa

La función de recompensa calcula a partir del estado actual del agente, y de la acción a tomar, una magnitud que indica qué tan bien está realizando el agente su tarea. Es la única realimentación que tiene el agente sobre su labor y es la única información disponible para que los algoritmos de aprendizaje reforzado decidan cómo deben mejorar esa recompensa.

Al maximizar las recompensas acumuladas, el agente puede determinar qué acciones son más beneficiosas para alcanzar objetivos a largo plazo. Por lo tanto, el diseño de una función de recompensa adecuada es esencial, ya que moldea el comportamiento del agente e impulsa el proceso de aprendizaje hacia los resultados deseados [13].

Una función de recompensa bien definida no solo anima al agente a alcanzar objetivos específicos, sino que también garantiza que el proceso de aprendizaje se mantenga estable y eficiente. Si la señal de recompensa es escasa o está mal definida, el agente tendrá dificultades para identificar patrones significativos a partir de sus interacciones, lo que conduce a comportamientos no deseados [13].

Procesos de decisión de Markov (MDP)

De manera simplificada, un *MDP* es un formalismo para modelar un sistema en un estado específico de manera que la probabilidad de pasar a un nuevo estado depende únicamente del estado anterior del sistema y de la acción tomada [12].

De manera general, en un *MDP* la dinámica del sistema se determina con las probabilidades de transición de un estado s_k en el instante k a un estado s_{k+1} en el instante $k + 1$ y con la acción a_k :

$$P(s', s, a) = Pr(s_{k+1} = s' \mid s_k = s, a_k = a) \quad (2.10)$$

Los algoritmos clásicos de RL procuran aprender de la experiencia estas probabilidades y con ellas intentar maximizar la recompensa esperada en una sucesión óptima de estados y acciones. Este proceso es en general complejo, y por ello las técnicas modernas evitan tener que aprender la dinámica del sistema de forma explícita, y buscan formas en que solo implícitamente estas probabilidades se aprendan [12].

Paso

Un paso (*step*) en el área de RL define la unidad básica de interacción entre un agente y su entorno. Tomando prestada la terminología de los sistemas de control, un paso se refiere a un único ciclo dentro del bucle de RL, como se muestra en la figura 2.4. Además, se describe en RL [13], el concepto, donde en cada paso, el agente realiza una acción en el entorno, observa el estado resultante y recibe una señal de recompensa, por lo que en un paso ocurre el intercambio de información entre los principales actores del proceso [12].

RL con redes neuronales

Dado que las funciones de probabilidad que representan la dinámica del sistema, las políticas o incluso, en ocasiones, las mismas funciones de recompensa, deben aprenderse a partir de la experiencia del agente interactuando con su entorno, es natural que para estos procesos de aprendizaje se utilicen redes neuronales artificiales (RNA) como modelos de aproximación de esas funciones, aprovechando sus propiedades de aproximadores universales [13].

Las técnicas de aprendizaje reforzado que usan redes neuronales proponen entonces estrategias de recolección de los datos con los que se entrenará la red, los valores de referencia o funciones de pérdida que se utilizan para entrenar esas redes, y las estrategias de cómo usar los aprendizajes parciales mientras las redes aprenden en todo el proceso de interacción [13].

Como no se conoce lo que debe hacer un agente para mejorar sus expectativas de recompensa, se convierte en un reto plantear los problemas de entrenamiento de las redes neuronales, utilizando únicamente las recompensas o castigos a los comportamientos.

Etapas de exploración y explotación

Durante el entrenamiento de un agente por medio de aprendizaje reforzado, se debe asegurar mantener un balance entre las llamadas etapas de exploración y explotación [13].

La etapa de exploración incentiva al agente a tomar riesgos y elegir acciones aleatorias, solo para adquirir la experiencia de nuevas posibilidades de obtener mejores recompensas. En la etapa de explotación, el agente usa la política que ha aprendido previamente para determinar qué acciones tomar, con el riesgo de que sea muy conservador y no tome riesgos para aprender nuevas cosas, limitándolo a repetir experiencias que ya conoce [13].

Los algoritmos o métodos de RL usualmente aplican la etapa de exploración al inicio de los entrenamientos de modelos para exponer al agente a situaciones que signifiquen una mayor recompensa. Con el avance del entrenamiento, se utilizan cada vez más las etapas de explotación, de modo que el agente aplica cada vez más las estrategias aprendidas para obtener su recompensa.

Una de las técnicas más utilizadas para lograr el balance entre la exploración y explotación en un entrenamiento se conoce como avaro- ε (*greedy- ε*). La técnica trabaja con una selección aleatoria, tal que con una probabilidad ε , el agente toma acciones aleatorias y con probabilidad $1 - \varepsilon$ toma acciones dadas por la política aprendida [13].

El efecto del cambio de ε se ejemplifica en la figura 2.5, donde se aprecia la afectación de la selección de ε : un $\varepsilon = 0$ (curva verde) provoca solo acciones aprendidas que no convergen a una recompensa máxima, mientras que el caso de $\varepsilon = 0,01$ y $\varepsilon = 0,1$ (curvas roja y negra, respectivamente) la exploración en ambas etapas resulta en una mejora progresiva de la recompensa. La velocidad de la convergencia hacia el máximo, varía asociado a qué tanto se le permite al agente explorar [13].

Categorización del RL

El RL dirige a los agentes a mejorar la recompensa en función de la política establecida para el control. Dada esa búsqueda del mejor desempeño en los entornos, se han propuesto numerosas técnicas o algoritmos para el RL, lo que se resume en la figura 2.6. Las categorías principales son: el RL basado en modelos y el libre de modelos. De igual forma se cuenta con el aprendizaje reforzado profundo (DRL), una combinación y reestructuración de métodos de cada subdivisión basada principalmente en las redes neuronales artificiales (RNA) con hasta cientos o miles de capas [12].

Cada método cuenta con sus premisas que los hacen adecuados para sistemas o entornos particulares, de manera que la identificación de los factores clave del entorno a controlar y su correcta caracterización permiten dirigir la selección de los métodos de RL. En esta ocasión se describen los algoritmos que se seleccionaron como posibles candidatos para el controlador del entorno PAMH.

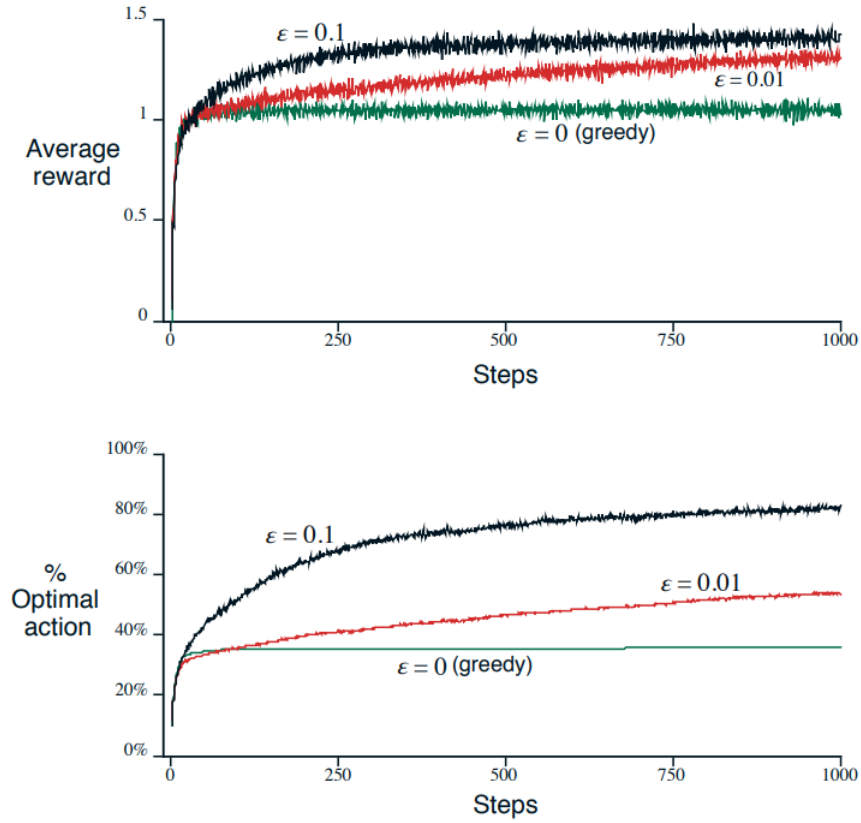


Figura 2.5: Efecto de la selección de la probabilidad ϵ en el desempeño del agente. Fuente [13].

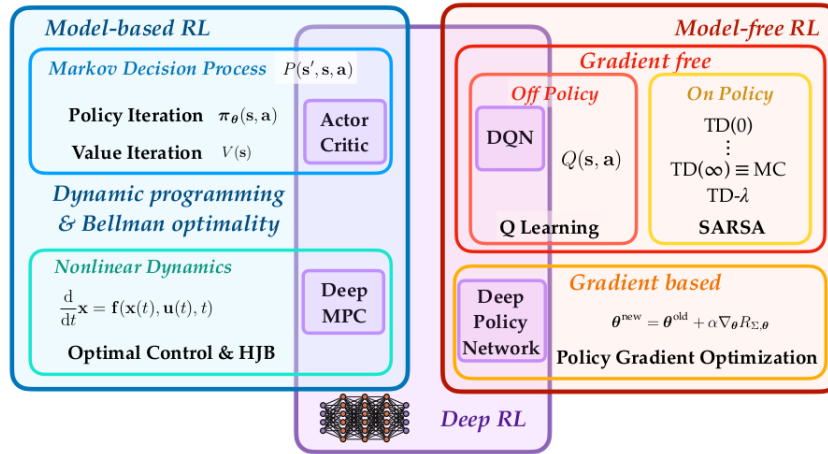


Figura 2.6: Resumen de categorización del RL [12].

2.3.2. Deep Q-Network (DQN)

El algoritmo (*Deep Q-Learning*, DQL) es un método de aprendizaje reforzado introducido por la compañía DeepMind Technologies (hoy Google DeepMind) en el año 2013. El DQL usa RNA para aproximar los llamados valores Q, por lo que se suele definir al método como DQN (*Deep Q-Network*). El algoritmo de funcionamiento general del método se muestra en el apéndice A [14].

Los valores Q provienen del algoritmo clásico de aprendizaje Q (Q -learning). Como se observa en la figura 2.6, se trata de un método fuera de política (*off-policy*), de manera que el aprendizaje de un agente se realiza mediante la exploración de acciones que no siguen la política que haya aprendido el sistema [12][15]. Se define el valor $V(s)$ en un estado s como la recompensa esperada en el estado s , si se siguiera una política óptima a partir de allí, o en otras palabras, la mejor recompensa esperable a partir del estado s . El valor Q es similar, y se relaciona con el valor $V(s)$ con:

$$V(s) = \max_a Q(s, a) \quad (2.11)$$

Es decir, el valor Q representa la recompensa total esperada al tomar una acción en un estado dado, siguiendo una política óptima a partir de allí.

Aprendizaje Q usa la técnica avaro- ε , de manera que con probabilidad ε el algoritmo utiliza una acción a_t aleatoria del espacio de acciones (exploración). Caso contrario, con probabilidad $1 - \varepsilon$ se toma la acción directa de la respuesta de la RNA (explotación) con:

$$a_t = \arg \max_a Q^*(\phi(s_t), a; \theta) \quad (2.12)$$

donde $\phi(s_t)$ es un mapeo del estado s_t a un espacio de mayor dimensión y θ corresponde a los pesos de la estructura de la red neuronal.

Las acciones y estados tomados se guardan en lotes para la valoración de sus recompensas respecto a la estimación de valores del lote, donde se les calcula una pérdida $L(\theta)$ para evaluar su desempeño con [14]:

$$target_j = \begin{cases} r_j & \text{para estado terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{para el resto } \phi_{j+1} \end{cases} \quad (2.13)$$

$$L(\theta) = E [(target_j - Q(\phi_j, a_j; \theta))^2] \quad (2.14)$$

donde $target_j$ es la referencia objetivo del lote anterior y el término $Q(\phi_j, a_j; \theta)$ corresponde a la predicción. Desde este punto es necesario aplicar el descenso de gradiente a la función de pérdida y actualizar los valores Q . Una cantidad N de iteraciones aseguran la convergencia del modelo a los valores deseados y por ende, comportamientos deseados del entorno [12][14].

2.3.3. Optimización de política próxima (PPO)

Se trata de un método basado en política (*on-policy*) propuesto en 2017 por la empresa OpenAI que aprende directamente a predecir la acción para espacios de estado discretos y continuos. Se busca cumplir una política definida y optimizarla poco a poco con actualizaciones. El método nace con la necesidad de utilizar un algoritmo de mayor accesibilidad en su implementación, al contrario de métodos similares pero de mayor complejidad, como el TRPO (*trust region policy optimization*). El algoritmo del método PPO se muestra en el apéndice B [16].

Entre las funciones clave del método PPO se encuentra la razón de los pesos de la política actual respecto a la anterior, con base en la función de política (2.9):

$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_\theta(a_{t_{old}} | s_t)} \quad (2.15)$$

que evidencia el cambio respecto a la política anterior y su valoración por medio de la función de pérdida del aprendizaje:

$$L^{CLIP}(\theta) = \mathbf{E}_t [\min(r_t(\theta) \mathbf{A}_t, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon) \mathbf{A}_t)] \quad (2.16)$$

donde ε es un hiperparámetro del algoritmo que asegura que el cambio a la política anterior no sea muy abrupto para el proceso y el vector \mathbf{A}_t corresponde al estimador de la función de ventaja, obtenido mediante la diferencia [16]:

$$\mathbf{A}_t = Q(s, a) - V(s) \quad (2.17)$$

lo cual genera la recompensa “extra” que se puede obtener al tomar las acciones $V(s)$ sobre los valores $Q(s, a)$ [17].

En suma, a través de un número N de iteraciones, el modelo pule poco a poco la política que va aprendiendo con cada cálculo de la pérdida (2.16) y se actualizan los parámetros θ de la RNA que convergen durante el proceso para alcanzar la política óptima y maximizan la expectativa de recompensa [12].

2.4. Métricas de evaluación

2.4.1. Función de recompensa

La función de recompensa sirve de guía a los agentes para decidir que acción tomar. Por consiguiente y de acuerdo con el comportamiento deseado, una función de recompensa engloba el objetivo, las preferencias y las restricciones que presenta la tarea, usualmente adquiriendo la información desde las observaciones al entorno e interpretándolas con base en el comportamiento deseado.

Por lo tanto a la hora de diseñar un sistema de RL, crear una función de recompensa y mantener el monitoreo en la misma, permite interpretar el comportamiento el agente y evaluar su desempeño, como en el ejemplo del caso en la figura 2.5 con la recompensa de cada valor de ε seleccionado [12].

2.4.2. Pérdida del modelo

El valor de pérdida (*loss*) en RL, como en el caso de la ecuación (2.16), suele obtenerse de los cálculos de los gradientes o procesos de optimización, en donde una pérdida baja en RL

no garantiza necesariamente un rendimiento óptimo. Principalmente indica que el agente está aprendiendo y mejorando su política basándose en la estructura de recompensas definida y en el algoritmo elegido [12].

La principal métrica en RL suele ser la recompensa obtenida por el agente, dado que refleja directamente lo bien que el agente ha aprendido a maximizar sus recompensas, pero la pérdida también desempeña un papel crucial a la hora de guiar el proceso de aprendizaje para la actualización de políticas y funciones de valor.

2.4.3. Error de posición angular

Todo problema que incorpora secciones del control automático es evaluado mediante la respuesta a una entrada conocida a la que se le mide el error de estado estacionario, el tiempo de estabilización, el porcentaje de sobreimpulso, entre otros [9][10].

Todo el proceso debe responder a una exigencia de punto de operación premeditado, en este caso un porcentaje de error de estado estacionario menor al 10 %.

Capítulo 3

Control de PAMH basado en aprendizaje reforzado

Para obtener un modelo controlador de una planta como el péndulo amortiguado de motor con hélice, es necesario definir el modelo de RL a utilizar, el plan de entrenamiento del modelo y la función de recompensa que conduzca al comportamiento deseado del agente.

3.1. Selección de modelos de RL

En primera instancia se realiza una investigación bibliográfica respecto a los métodos de RL comunmente utilizados para el control de entornos o sistemas similares a la planta PAMH, al igual que una búsqueda de repositorios o publicaciones realizadas para cada método de RL encontrado, de manera que cada uno de los métodos contara con un respaldo para la evaluación de los criterios seleccionados para la matriz de Pugh, mostrada la tabla 3.1.

Tabla 3.1: Matriz de Pugh de los métodos de RL candidatos.

Criterios	Peso	Alternativas			
		DQN	PPO	DDPG	SAC
Fiabilidad de control del PAMH	4	0	+1	+1	+1
Recursos computacionales	3,5	+1	+1	0	-1
Tiempo de desarrollo	3	+1	+1	0	0
Código existente	2,5	+1	+1	0	0
Optimización	2	+1	+1	0	0
Tiempo de entrenamiento	1,5	-1	+1	0	+1
Innovación	1	-1	0	+1	+1

Suma general	8,5	16,5	5,0	3.0
Ranking	2.º	1.º	3.º	4.º

La revisión de material respecto al algoritmo DQN se encuentra en el documento base del método [14] publicado en el año 2013 y los criterios o consideraciones importantes al respecto se encuentran en [15] y [18], como el problema de convergencia con espacios de acciones continuos [19]. En resumen, el método DQN cuenta con amplio respaldo de ejemplos de implementación en RL y presenta entrenamientos con resultados eficientes para espacios de acciones discretos y soluciones discretizadas.

En cuanto al método PPO, se consultó el documento base del método [16] publicado en el año 2017 y de igual forma, repositorios y artículos sobre sus casos de aplicación, donde se demuestra su eficiencia al trabajar tanto con espacios de acciones discretos como continuos [20] y [21].

El caso del método gradiente de política determinista profunda (DDPG, por sus siglas en inglés) publicado en el año 2019, responde a una solución de los problemas del DQN con espacios continuos, en ocasiones conocido como DQN para espacios continuos. El algoritmo utiliza lógica del conocido DQL e implementaciones de políticas de igual forma [22].

También se tiene el método actor-crítico blando (SAC, por sus siglas en inglés) publicado en el año 2018, un algoritmo basado en RL de máxima entropía, pues el componente actor busca maximizar la recompensa esperada junto con la entropía, por lo que la convergencia a un comportamiento deseado es caracterizado por pasos “suaves” en lugar de saltos abruptos en acciones [23].

El factor del recurso computacional requerido define el tiempo de entrenamiento o en algunos casos, la posibilidad de implementación del método. El DQN dispone de amplio repositorio de ejemplos, pero se trata del método de mayor antigüedad, por lo que no presenta el mismo enfoque de disminución de tiempos a costo de la capacidad de cálculo como si lo hicieron DDPG y SAC [22] [23].

Por otro lado, los métodos DDPG y SAC al ser nuevos en comparación con los otros, no cuentan con el amplio registro de repositorios de implementación y pasos extra de optimización, mientras si cuentan con extensos algoritmos de funcionamiento, por lo que su tiempo de desarrollo se ve afectado.

El método PPO presenta las mejores cualidades para ser utilizado como algoritmo entrenador de un controlador para el entorno PAMH, descrito en el capítulo 2. Como segunda posición se tiene el método DQN, por lo que se puede seleccionar de acuerdo con lo recomendado en las publicaciones referentes a su uso con discretización del espacio de acciones continuo.

3.2. Preparación de entornos virtuales

Dada la complejidad de los métodos de RL utilizados, el primer paso consiste en una implementación con entornos virtuales conocidos. La plataforma Gymnasium de Farama

Fundation [24] ofrece una colección de entornos de código abierto disponibles para experimentación con estrategias de control. Los entornos consisten en simulaciones físicas de entornos particulares como péndulos simples y dobles, con actuadores y un conjunto de acciones predefinidas [24].

Luego del estudio de los algoritmos pertinentes para la aplicación del RL mediante PPO y DQN [16] [14], se aplican ajustes para funcionar, en primera fase, con entornos de acciones discretas, como es el caso del problema de control del péndulo invertido (*CartPole*) [24].

El segundo paso es adaptar el código para trabajar con entornos con características y mediciones de valores continuos. Como caso de estudio se usa el control del otro péndulo invertido (*Pendulum*), en el cual se realizan las primeras pruebas de ajuste de ángulo objetivo, dado que el objetivo base del entorno es mantener el ángulo superior estático (0°). Por lo tanto, el primer ajuste al método fue la implementación del error de ángulo:

$$E_\theta = |\theta - \theta_{\text{objetivo}}| \quad (3.1)$$

en lugar de únicamente el componente θ del péndulo en su función de recompensas, heredada del código base de Gymnasium [24].

Para facilidad de acoplamiento de los modelos experimentales anteriores al caso concreto de PAMH, se adapta el modelo mimetizador del proyecto previo [6] a una interfaz de tipo Gymnasium, de manera que el modelo imitador del PAMH trabaje de forma similar a los modelos disponibles de RL que se usan en Gymnasium.

En las mediciones u observaciones al entorno simulado o real PAMH, hay un único valor observable: el ángulo del péndulo de la planta. Se adicionan parámetros de información para el procesamiento de la red neuronal para complementar el estado con aproximaciones de la primera derivada (como velocidad angular) de la forma:

$$\dot{\theta} = \theta_t - \theta_{t-1} \quad (3.2)$$

y la segunda derivada (como aceleración):

$$\ddot{\theta} = \dot{\theta}_t - \dot{\theta}_{t-1} \quad (3.3)$$

ambos casos basados en las diferencias entre los valores medidos en la actual iteración t y la anterior iteración $t - 1$.

Se agrega además a las observaciones el valor del ángulo objetivo θ_{objetivo} , con la intención de que la red pueda aprender a inferir diferencias de comportamiento de la planta, dependientes del objetivo que esta deba alcanzar; pues no es lo mismo controlar para llegar a 10° que a 90° . Por lo tanto, el vector utilizado de observación de la planta es:

$$obs_n = [\theta, \dot{\theta}, \ddot{\theta}, \theta_{\text{objetivo}}] \quad (3.4)$$

lo que corresponde al mapeo $\phi(s)$ de una dimensión a cuatro dimensiones, propuesto en este trabajo para cumplir la función introducida de forma general en la ecuación (2.12).

3.3. Parametrización de los modelos

Uno de los puntos clave del trabajo realizado es el diseño de la función de recompensa enfocada en guiar al comportamiento deseado.

De igual forma, se presentan a continuación las configuraciones de los dos métodos seleccionados DQN y PPO.

3.3.1. Función de recompensa

La función de recompensa condiciona el comportamiento del agente por su relación directa con las características deseadas respecto a su comportamiento en el entorno.

Con base en los fundamentos relacionados a la planta PAMH, el problema definido para el agente es mantener un ángulo constante, indicado con anterioridad, además de alcanzar una velocidad angular baja, cuando se alcance el ángulo final.

Por lo tanto, se propone una ecuación que recompense alcanzar el ángulo deseado y caso contrario, castigar un ángulo diferente al deseado y una velocidad angular alta, pues este último indicaría que no se estabiliza el péndulo. La función propuesta se muestra en el algoritmo 1.

En este caso la función de recompensa se centra en dos de las cuatro observaciones del entorno: el ángulo del péndulo θ y la velocidad angular $\dot{\theta}$. El objetivo de este caso es alcanzar el ángulo objetivo y que el péndulo permanezca estático allí. Por lo tanto, primero se verifica la escala del ángulo medido θ para evitar problemas de precisión por redundancia de ángulos fuera de rango $[-\pi, \pi]$. Luego se calcula el error angular que es potenciado al cuadrado para dar mayor castigo a los valores lejanos al $\theta_{objetivo}$. La baja magnitud de $\dot{\theta}$ requiere aumentar cien veces su valor para afectar la comparación con el ángulo y potenciarla para mayor castigo por movimientos rápidos, lo que se determina experimentalmente.

El valor de R_θ busca recompensar la cercanía con el objetivo, pero solo al máximo ($R_\theta = 2$) si la velocidad $\dot{\theta}$ mantiene un valor cercano a cero, aproximación a péndulo estático.

El componente $E_{a_{t-1}}$ es el error por la acción anterior que produjo la observación obs_n y castiga la generación de señal PWM por encima del 0,25, valor elevado para la planta real, por lo que se debe evitar. Si la señal de acción se mantiene en el rango con límite inferior 0,0 y límite superior 0,25, no hay castigo.

La salida del algoritmo selecciona el mayor castigo entre el C_θ y $E_{a_{t-1}}$; si la señal de acción se encuentra en rango, la recompensa solo será el error del ángulo con la recompensa por cercanía con el objetivo.

La figura 3.1 muestra la función de recompensa con $\theta_{objetivo} = 45^\circ$ con un rango de -120° a 120° y mayor detalle de la curva alrededor del objetivo se ilustra en la figura 3.2. Como se observa, a mayor distancia con el objetivo, mayor castigo, como el caso de un $\theta \approx 100^\circ$

Algoritmo 1 Función de recompensa

- 1: Recibe los componentes de la observación $\theta, \dot{\theta}$.
- 2: Recibe la señal de acción PWM anterior a_{t-1} .
- 3: Recibe el ángulo objetivo $\theta_{objetivo}$.
- 4: Asegura el valor θ en radianes y en el rango $[-\pi, \pi]$.
- 5: Cálculo del error:

$$E_{\theta} = |\theta - \theta_{objetivo}|$$

- 6: Potenciar y escalar el comportamiento no deseado con el costo por error angular C_{θ} y el costo por velocidad angular $C_{\dot{\theta}}$:

$$C_{\theta} = \theta_{error}^2$$

$$C_{\dot{\theta}} = 100 \cdot \dot{\theta}^2$$

- 7: Recompensa por comportamiento deseado:

$$R_{\theta} = 2 \cdot (e^{-C_{\theta}/\sigma_r} (1 - |C_{\dot{\theta}}|))$$

- 8: Recompensa y castigo por señal de acción deseada:

$$E_{a_{t-1}} = \begin{cases} 0 & \text{para acción } 0 < |a_{t-1}| < 0,25 \\ 20^{|a_{t-1}-0,25|} & \text{para el resto de } a_{t-1} \end{cases}$$

- 9: La señal de recompensa resultante:

$$R = \min [-C_{\theta}, -E_{a_{t-1}}] + R_{\theta}$$

con una recompensa de aproximadamente -1 unidades, mientras el castigo tiende a cero si se acerca al $\theta_{objetivo}$, punto en que se influye el componente R_{θ} , afectando un caso de $\theta \approx 43^\circ$ con recompensa de 1,5 unidades aproximadamente.

Cabe resaltar que para la definición del último punto de la función de recompensas expuesta, se requirió de experimentación con numerosos entrenamientos de los modelos, pues un cambio como la aplicación de una resta lineal

$$R = -C_{\theta} - E_{a_{t-1}} + R_{\theta} \quad (3.5)$$

o prescindir del componente $E_{a_{t-1}}$, da libertad al modelo para explorar sus límites de acción. La aplicación de la técnica de moldear la recompensa (*reward shaping*), impactan sensiblemente el desempeño del agente, en este caso seleccionando la opción del algoritmo 1 por mantener el nivel del resultado del proceso de entrenamiento estable [25].

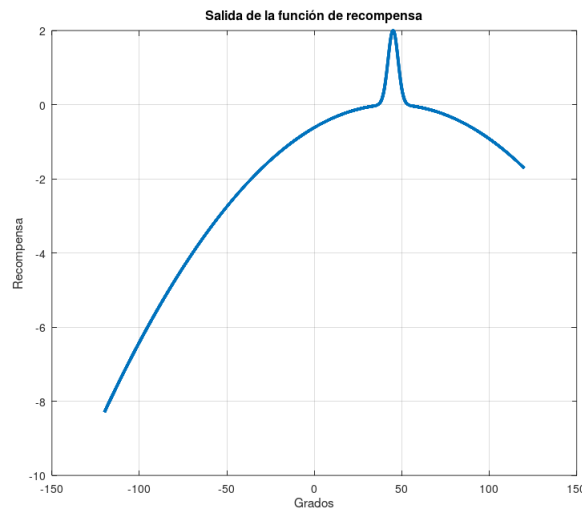


Figura 3.1: Función de recompensa con $\theta_{objetivo} = 45^\circ$.

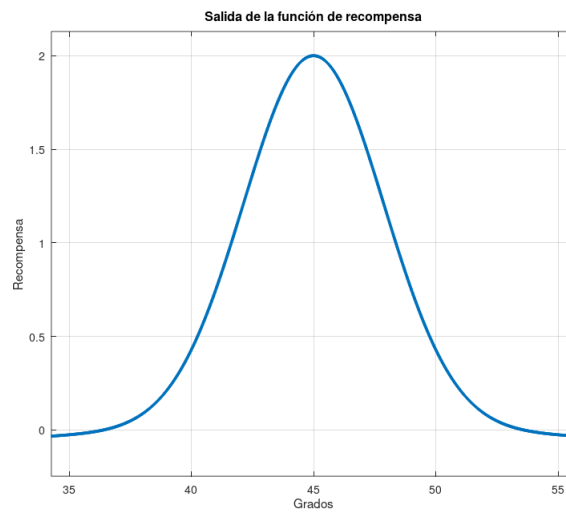


Figura 3.2: $\theta_{correcto}$ con $\theta_{objetivo} = 45^\circ$.

3.3.2. Estructura de las RNA en DQN y PPO

La estructura general de las RNA en ambos modelos utiliza una capa de entrada definida por el número de observaciones del entorno, anteriormente definido como obs_n en la ecuación (3.4), una capa de salida definida por el número de acciones posibles por método, descritos en las siguientes secciones, y un conjunto de tres capas densas (completamente conectadas) con número de neuronas 64, 128, 64, respectivamente, para requerir un aumento de dimensión y acceso a procesamiento más complejos en la RNA, esto definido para cada método [12].

Como capas de activación, se utiliza ReLU (*rectified linear unit*) por la condición del espacio de acciones base del entorno: valores de PWM en el rango de 0,0 a 1,0, de manera que los valores positivos sean recibidos de manera lineal y los negativos ignorados [12].

3.3.3. DQN

Discretización

Como se mencionó en el capítulo 2, el método DQN se utiliza en el control de sistemas; sin embargo, la literatura indica que DQN tiene problemas en el caso de entornos con espacios de acciones continuos [19].

Por lo tanto, se discretizan las acciones del entorno, que oficialmente trabaja en el rango de 0,0 a 1,0 de PWM. De acuerdo con [6], el límite superior del espacio de acciones se fija por seguridad en 0,25 de PWM para cuidar la integridad del equipo real, por lo que se respetó en las pruebas virtuales, además de que la RNAM no conoce el comportamiento de la planta real fuera de ese rango.

Se selecciona una cantidad de 10 posibles acciones discretas, producto de la experimentación, lo cual responde a la función:

$$a_d = \lfloor 36 \cdot a \rfloor \quad (3.6)$$

para el proceso de discretización y la función:

$$a = \frac{a_d}{36} \quad (3.7)$$

para el proceso de desdiscretización. Esto deja un espacio de acciones discretas mostradas en la tabla 3.2.

Tabla 3.2: Equivalencia de valores continuos y discretos seleccionados para el método DQN.

Tipo	Acción equivalente									
Continua (aproximada)	0.0	0.03	0.06	0.08	0.11	0.14	0.17	0.19	0.22	0.25
Discreta	0	1	2	3	4	5	6	7	8	9

RNA para DQN

Tomando la base ya expuesta, la RNA utilizada para el método DQN parte de las estructuras encontradas en repositorios como [18], utilizando 128 neuronas en la capa densa central, a la que se le suman una capa densa previa y una capa densa posterior, ambas de 64 neuronas para el aumento y disminución de la dimensionalidad. Por último, se acopla la capa de salida con las 10 posibles salidas de valores Q mencionadas. El resultado como tal se muestra en la figura 3.3, de manera que se obtiene un vector Q_n con los valores Q para cada posible acción.

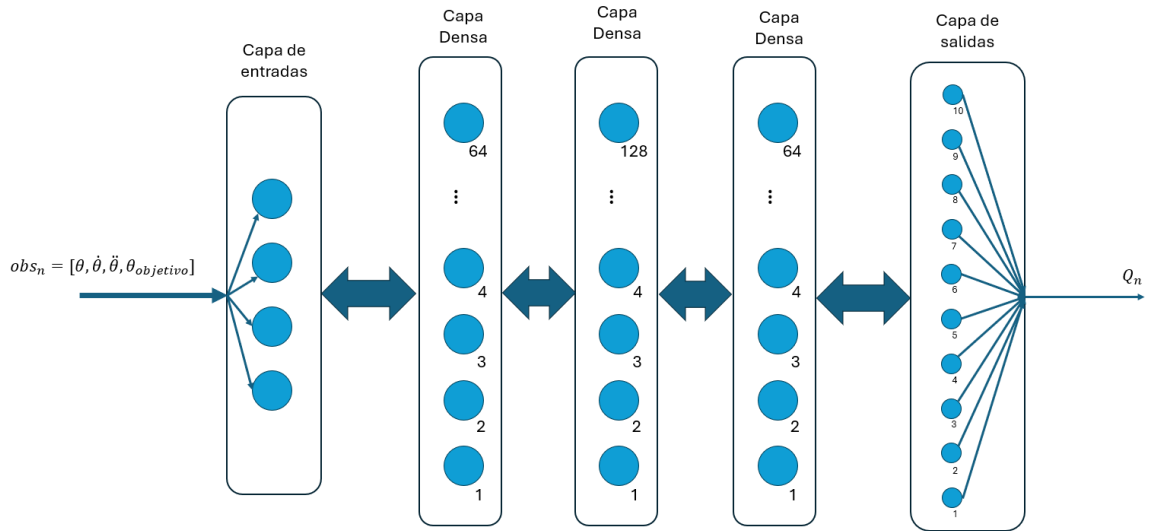


Figura 3.3: Estructura de la RNA utilizada en el método DQN.

Plan de entrenamiento

El esquema original de entrenamiento de DQN se modifica en la selección de la acción; utilizando avaro- ε , la etapa de exploración implica la generación de un valor de ruido n_{ruido} que es sumado a la acción discreta dictada por la RNA que aproxima (2.12), obteniendo:

$$a_{t_{ruido}} = a_t + n_{ruido} \quad (3.8)$$

donde n_{ruido} varía en principio en un rango con magnitud máxima de 6 y mínima 0. Luego al alcanzar la mitad del tiempo de entrenamiento, la magnitud máxima del ruido se reduce a 4 y luego de tres cuartos del tiempo se reduce a 2. Por lo tanto, el avaro- ε reduce las probabilidades de adición de ruido y al mismo tiempo la magnitud del ruido n_{ruido} se disminuye con el avance del entrenamiento, pasando a un método de explotación de lo aprendido.

3.3.4. PPO

RNA para PPO

El caso del método PPO cuenta con la misma base de RNA mencionada anteriormente en suma a lo encontrado en los repositorios que implementan el método como [26]. Se incorporan tres capas densas compuestas por 64, 128 y 64 neuronas respectivamente, a lo que se suma la capa de salida a una neurona única para el valor de PWM a utilizar. El resultado como tal se muestra en la figura 3.4.

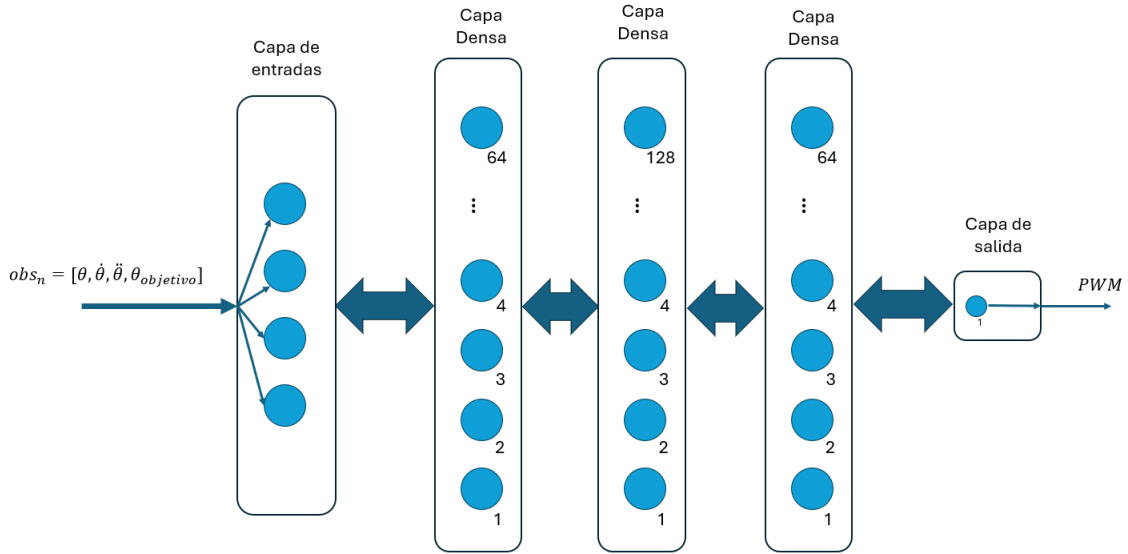


Figura 3.4: Estructura de la RNA utilizada en el método PPO.

Plan de entrenamiento

El algoritmo PPO base define su etapa de exploración con acciones aleatorias muestreadas a partir de una función de distribución normal centrada en la acción tomada por la RNA. En este caso se modificó el procedimiento para cumplir con las siguientes rutinas; se define una muestra de ruido de la distribución pero se fija su uso en una cantidad determinada de iteraciones, en un principio 10, para ir disminuyendo una unidad cada doceava parte del tiempo de entrenamiento total, esto hasta mantener una muestra por iteración hacia el final del entrenamiento. Este cambio fue necesario para que el ruido agregado le permita al péndulo reaccionar, pues de otro modo la inercia del modelo elimina todo efecto de dicho ruido, que pretende forzar cierta exploración del espacio de estados.

Otra modificación propuesta es la disminución del valor de varianza inicial utilizado, $\sigma^2 = 0,01$ para crear la distribución normal que se muestrea en el punto anterior. La varianza disminuye con el mismo paso que la muestra estática mencionada, pero con una magnitud de 0,001, esto hasta llegar a un valor mínimo del $\sigma^2 = 0,001$.

Por último, en lugar de utilizar la muestra directa de la distribución como acción a_t , se considera como una muestra de ruido n_t sumada con la señal a_t de la RNA, de la siguiente manera:

$$a_n = a_t + n_t \quad (3.9)$$

resultado al que se le limita la respuesta respetando los valores de seguridad mencionados ($min = 0, max = 0,25$). Al resultado limitado a'_n se le sustrae el a_t y se le calcula la probabilidad de aparición del ruido n'_t en la distribución normal.

Todo los pasos anteriores ocurren durante las pruebas de observación al entorno y respuesta de la RNA, únicamente cambiando sus valores al cumplir el paso de una doceava parte del entrenamiento a la vez.

3.3.5. Procesamiento y recopilación de datos

En el presente proyecto se experimenta con los procesos de entrenamiento de modelos de RL para el control de la planta simulada PAMH, proceso que según se vió en la tabla 3.1, requiere recursos computacionales considerables. Los entrenamientos y pruebas se realizaron en un computador con procesador Intel *i7-7700HQ* de $2,80\text{ GHz}$ con 8 núcleos, con acceso a una unidad GPU de NVIDIA GeForce GTX 1050 con 4 GB de VRAM y mediante la herramienta *Google Colab* de la empresa Google, utilizando *Python 3 Google Compute Engine backend* con hasta 12 GB de RAM y 108 GB de almacenamiento.

Por otro lado, el protocolo de todos los procesos de entrenamiento realizados se concentran en la plataforma en línea *Weights & Biases* (W&B), servicio en línea orientado al aprendizaje automático.

3.4. Evaluación de los modelos

La evaluación del desempeño de los modelos se realiza con el promedio de la recompensa obtenida con el transcurso de los episodios en el entrenamiento de los modelos, donde se comparan diferentes resultados del promedio de la recompensa (diferentes entrenamientos) y se interpretan las aproximaciones a los valores más altos.

Luego de entrenado el modelo, se vuelve a montar el entorno con interfaz Gymnasium y se renderiza el comportamiento del PAMH en su versión virtual. En su segundo episodio se generan una curva de la respuesta angular del entorno respecto al tiempo, tomado directamente del sistema computacional. Con la curva de θ contra el tiempo en nanosegundos, se calcula el tiempo de subida, el tiempo de asentamiento y el sobreimpulso en la respuesta, esto procurando que el valor final del ángulo se encuentre en un rango de $\pm 10\%$ con diferencia al valor deseado $\theta_{objetivo}$.

Capítulo 4

Resultados y análisis

Este capítulo se presenta los resultados de la implementación de los modelos RL seleccionados para el control de la planta PAMH.

4.1. Implementación de métodos RL

Se ilustran a continuación los experimentos realizados con los entornos virtuales de problemas clásicos de control, como el del péndulo invertido.

Luego se presentan los resultados de la implementación de los modelos DQN y PPO en el control de la planta simulada PAMH.

4.1.1. Preparación del entorno virtual

Para el caso del DQN la principal fuente fue [18], donde se detalla el funcionamiento de cada sección de los algoritmos de trabajo de DQN.

El primer paso es montar el código base para el control del péndulo invertido con carrito (*CartPole*), el cual solo requiere de la instalación de la paquetería necesaria para trabajar con tensores y procesamiento mediante CUDA.

El resultado de la primera prueba de implementación se ilustra en la figura 4.1, donde el código base tiene como objetivo mantener el péndulo invertido apuntando hacia arriba el mayor tiempo posible. En este caso alcanzando las 500 unidades del valor máximo de duración del episodio y demostrando la eficacia del DQN en el entrenamiento, en donde en aproximadamente el episodio 450 de 600, el agente aprende la táctica adecuada para mantener el equilibrio del péndulo.

Como segundo paso se opta por el método PPO para trabajar con el péndulo invertido (*Pendulum*), en este punto se definen las gráficas de recompensa, pérdida, tiempo de la iteración y duración del episodio, todos como valores promedio en el entrenamiento y se

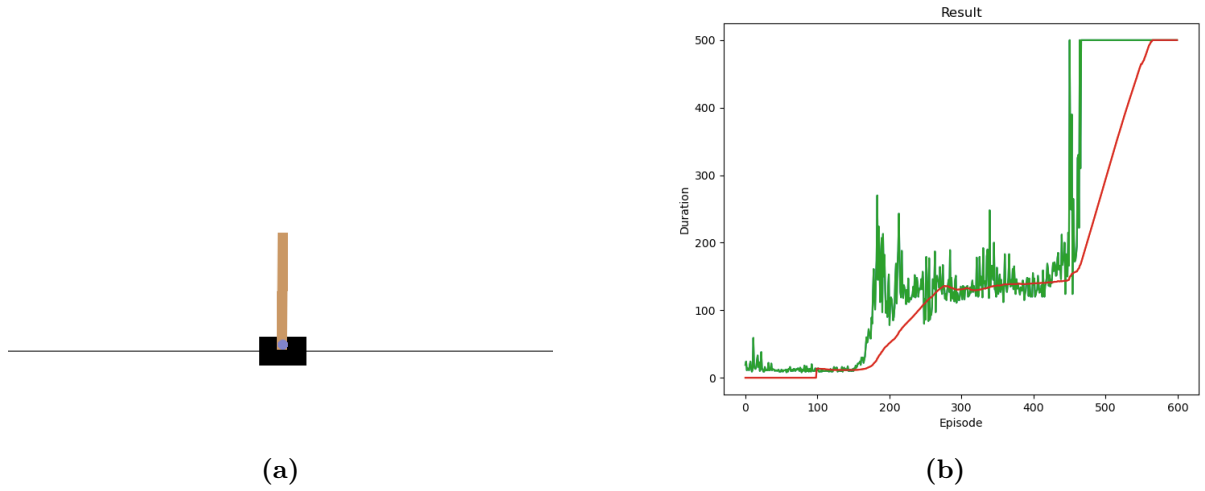


Figura 4.1: Duración del episodio en prueba de implementación DQN con entorno *CartPole*.

adiciona a la función de recompensa el factor de error angular E_θ de la ecuación (3.1), para indicar el $\theta_{objetivo}$. La prueba de implementación se muestra en la figura 4.2 con el resultado de la prueba, donde se muestra un comportamiento inesperado por un “atajo” descubierto por el agente, obteniendo una recompensa promedio aceptable (curva azul) pero terminando el episodio prematuramente (curva morada), por lo que el castigo es menor, caso que debe ser ajustado o cubierto mediante la función de recompensas.

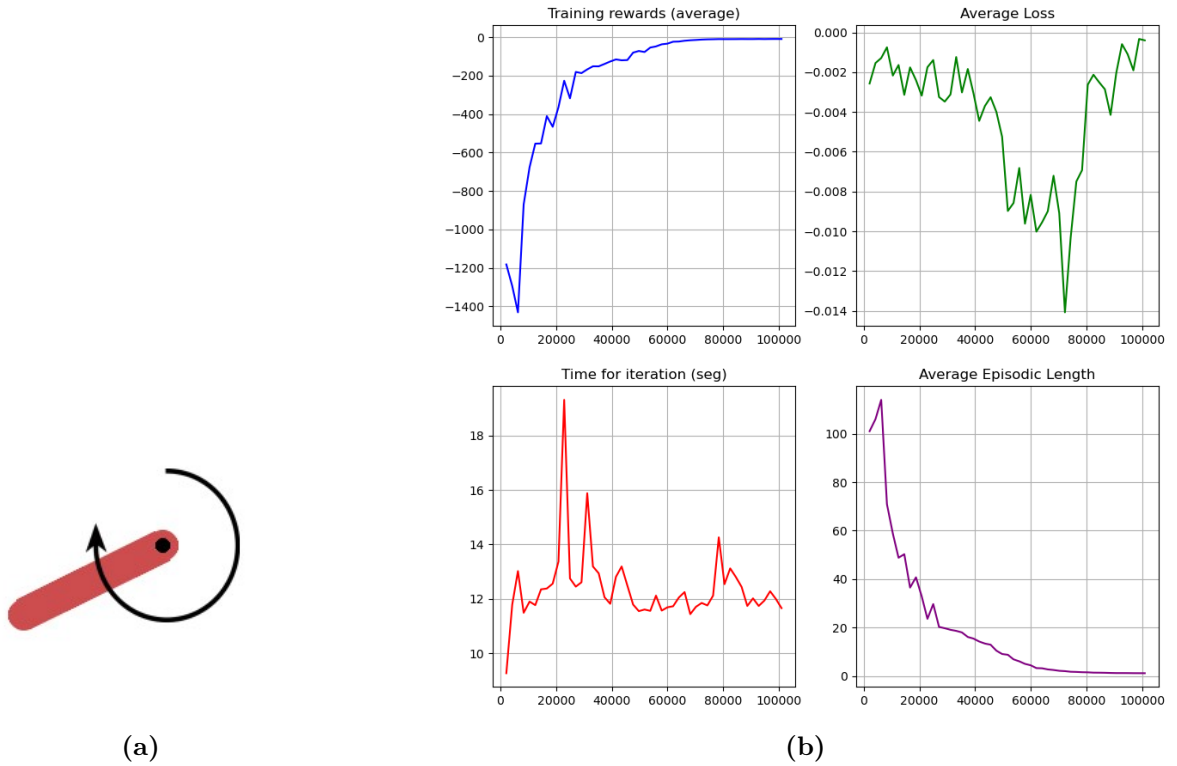


Figura 4.2: Proceso de entrenamiento con la implementación PPO con entorno *Pendulum*.

En este punto ya se cuenta con las bases necesarias del RL para la implementación de los métodos en el entorno virtual preparado con la RNAM del PAMH.

4.1.2. Modelo DQN

El código base del DQN tiene una implementación directa, pero el paso de discretización implicado en el enfoque mencionado a la RNAM del PAMH, aumenta el nivel del análisis y dificulta el proceso de entrenamiento. Una primera muestra de los resultados de los entrenamientos se ilustra en la figura 4.3a, donde se observa que la cualidad de discreto afecta a las recompensas obtenidas al generar saltos en las curvas, y una difícil aproximación al valor objetivo $\theta_{objetivo} = 45^\circ$ lo cual es un reflejo directo de la baja recompensa que se recibe; sin embargo, se logra acercarse al objetivo de manera prematura como se observa en la figura 4.4, indicando mejoría.

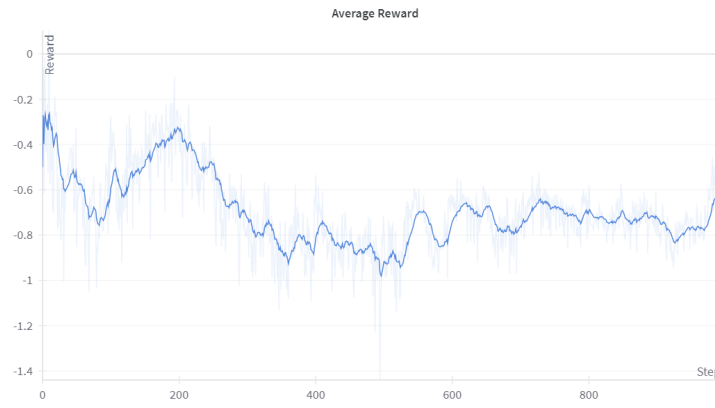
Al considerar la pérdida, se observa en la figura 4.3b, cómo los puntos de disminución de la pérdida coinciden con el aumento de recompensa, como el caso de los pasos (*steps*) aproximados a 200 y 900, lo que se interpreta como un modelo que aprende y en el caso de DQN, logra predecir un correcto valor Q que aumenta la recompensa. Sin embargo, también es posible interpretar que las predicciones vuelven a perder el rumbo deseado, pues en puntos como el paso aproximado a 500, la pérdida aumenta, indicando una pérdida de la predicción y por consiguiente, recompensa muy baja. Esto ocurre debido a que la componente exploratoria usada durante el entrenamiento desvía la atención del agente y lo lleva a condiciones donde no tiene experiencia para reaccionar adecuadamente.

Además, los comportamientos descritos pueden estar directamente relacionados el factor “discretización”, pues el modelo parece aprender una buena táctica en los puntos indicados, pero pierde el rumbo o se queda sin posibilidades para alcanzar el objetivo, por lo que puede estar relacionado con una decisión de no aumentar o dividir el PWM y pasarse de lejos el $\theta_{objetivo}$.

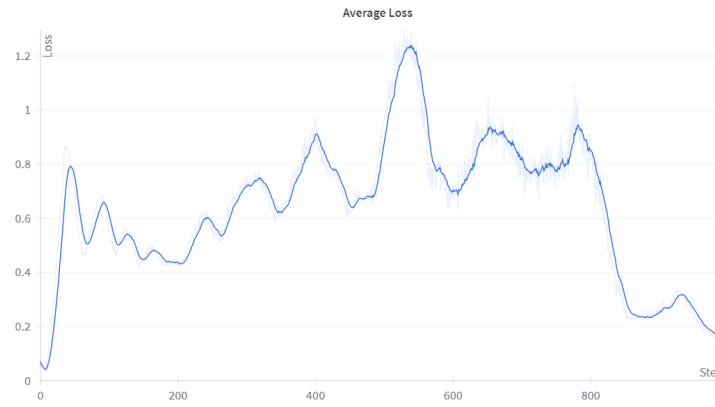
4.1.3. Modelo PPO

Luego de aplicar el algoritmo base PPO con los cambios resaltados en el capítulo 3, el proceso de entrenamiento demostró resultados prometedores desde la perspectiva del RL, ilustrados en la figura 4.5, donde se muestra el efecto del cambio del hiperparámetro varianza σ^2 en la curva, pues está directamente relacionada con el rango posible de valores aleatorios al que el agente se ve expuesto en la etapa de exploración. La curva rosa se expone a menos posibilidades de acción, pero la eficiencia del modelo permite que de igual forma converga a una recompensa positiva de aproximadamente 0,2, sin embargo, puede tratarse de un caso de éxito con poca experiencia, donde un pequeño desajuste podría desequilibrar por completo al modelo.

La curva verde con el hiperparámetro fijo en $\sigma^2 = 0,01$, muestra una convergencia lenta en comparación, apenas llegando a $-0,175$ aproximadamente, pues el “ruido” experimentado agrega complejidad a la búsqueda del $\theta_{objetivo} = 45^\circ$, requiriendo mayor tiempo de entrenamiento. Es posible proyectar un escenario donde a pesar del aumento de la recompensa, el agente nunca pueda llegar al valor deseado $\theta_{objetivo}$ o caso contrario, alcanzar el objetivo



(a) Recompensa promedio del modelo DQN. La línea tenue en el fondo representa la señal completa, y la línea oscura dicha señal filtrada para rescatar su tendencia.



(b) Pérdida promedio del modelo DQN. La línea tenue en el fondo representa la señal completa, y la línea oscura dicha señal filtrada para rescatar su tendencia.

Figura 4.3: Proceso de entrenamiento recompensa-pérdida de la implementación DQN con entorno PAMH.

con el modelo de mayor experiencia frente a desequilibrios.

La curva morada muestra una convergencia satisfactoria a una recompensa superior al 0,2 de la curva rosa, pero contando con una experiencia o política más completa de las técnicas para llegar al objetivo, esto gracias a la variación o barrido de valores de σ^2 , de modo que el ruido produce exploración suficiente para luego dar paso a la explotación de lo aprendido. Por lo tanto, representa el modelo mejor preparado.

También es posible observar en la figura 4.6 el valor final del ángulo por episodio, donde las tres curvas tienden al $\theta_{objetivo} = 45^\circ$, pero la curva rosa no exploró los ángulos lejanos y la curva verde requiere mayor estabilidad hacia el objetivo; los tres casos son proyecciones equivalentes a su recompensa promedio.

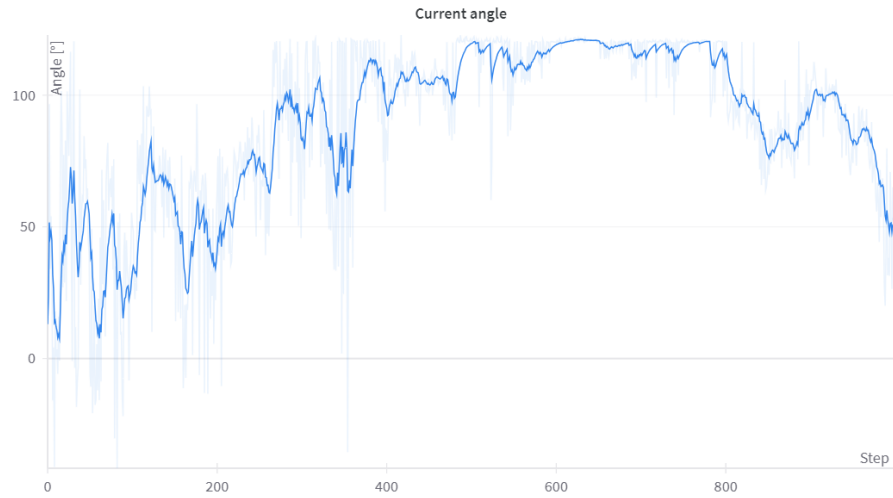


Figura 4.4: Últimos ángulos muestreados de los episodios del entrenamiento DQN con PAMH. La línea tenue en el fondo representa la señal completa, y la línea oscura dicha señal filtrada para rescatar su tendencia.

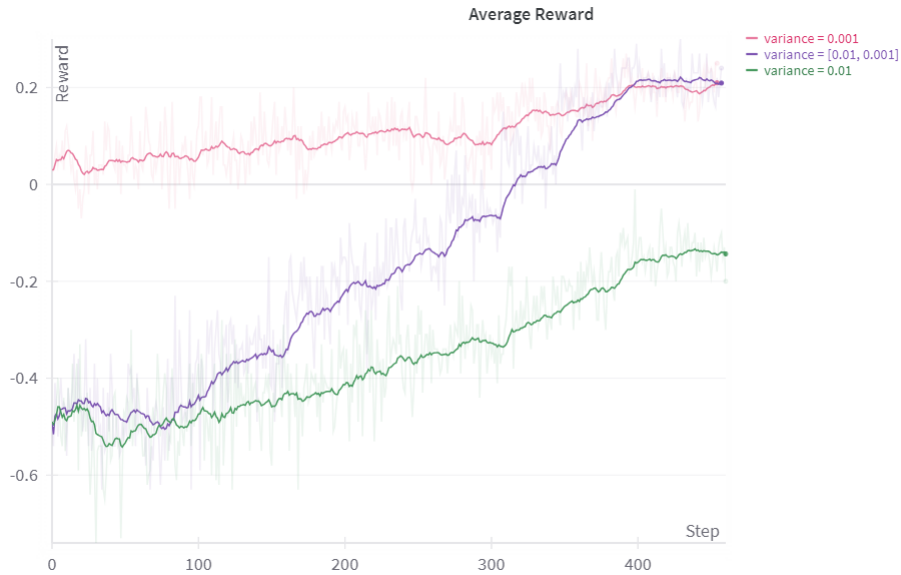


Figura 4.5: Recompensa promedio de los entrenamientos con diferentes valores de varianza σ^2 . La curva rosa contempla $\sigma^2 = 0,001$, la curva verde contempla $\sigma^2 = 0,01$ y la curva morada contempla el barrido en el rango de $\sigma^2 = [0,01, 0,001]$. Las líneas tenues en el fondo representan la señal completa, y las líneas oscuras dichas señales filtradas para rescatar sus tendencias.

4.2. Evaluación del desempeño de los métodos RL

La prueba final de evaluación requiere observar el efecto directo de la RNA entrenada frente al entorno sin perturbaciones o etapas de entrenamiento dictadas, únicamente la respuesta directa de la RNA.

Como se observó en la figura 4.3a, el método DQN no llega de manera certera al ángulo

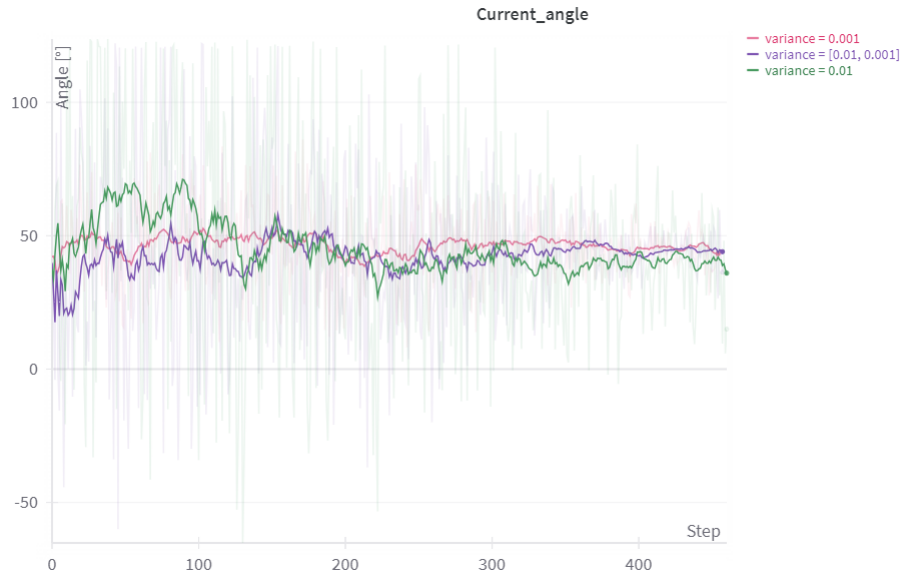


Figura 4.6: Últimos ángulos muestreados de los episodios en el entrenamiento. La curva rosa contempla $\sigma^2 = 0,001$, la curva verde contempla $\sigma^2 = 0,01$ y la curva morada contempla el barrido en el rango de $\sigma^2 = [0,01, 0,001]$. Las líneas tenues en el fondo representan la señal completa, y las líneas oscuras dichas señales filtradas para rescatar sus tendencias.

objetivo, por lo cual los parámetros de control no son medibles, sin embargo, muestra intentos de aprendizaje con el aumento de recompensa en algunos periodos y al alcanzar el objetivo, pero de forma casi inmediata se pierde el avance, por lo que su comportamiento es de un sistema no amortiguado como se muestra en la figura 4.7, donde se agrega la señal PWM resultante que el agente aplica para el control. El algoritmo no logra encontrar una política para determinar las acciones discretas que conducen al comportamiento objetivo.

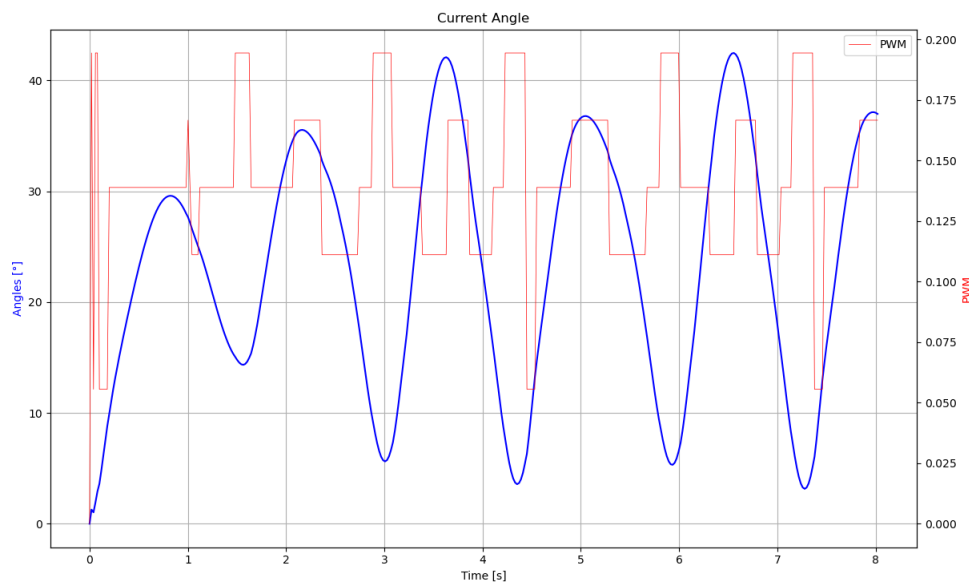


Figura 4.7: Curva de respuesta del modelo DQN ángulo respecto al tiempo.

En cuanto al modelo entrenado con PPO, se tiene la curva de la figura 4.8 donde se observan, a pesar del comportamiento durante el entrenamiento, el efecto de subamortiguado, pues la oscilación hacia el valor final es alta y requiere más de 14 s para alcanzar un estado de reposo, que también se trata de un valor incorrecto, aproximado a 29° . Además, los datos experimentales medidos fueron: tiempo de subida $t_r = 0,2603$ s, tiempo de estabilización $t_s = 15,78$ s y un sobreimpulso de $M = 75,66\%$.

Si bien estos valores son deficientes desde la perspectiva del control automático, se anota que estas métricas no han sido consideradas aun dentro de la función de recompensa, que en este caso solo incluye términos para el error angular y la velocidad terminal del péndulo. Las curvas de entrenamiento muestran que, incluso con la perturbación con el ruido del proceso de entrenamiento, el ángulo sí converge a los 45° , pero no se ha logrado encontrar la causa de por qué en la puesta a prueba del modelo aislado lleva a otro ángulo.

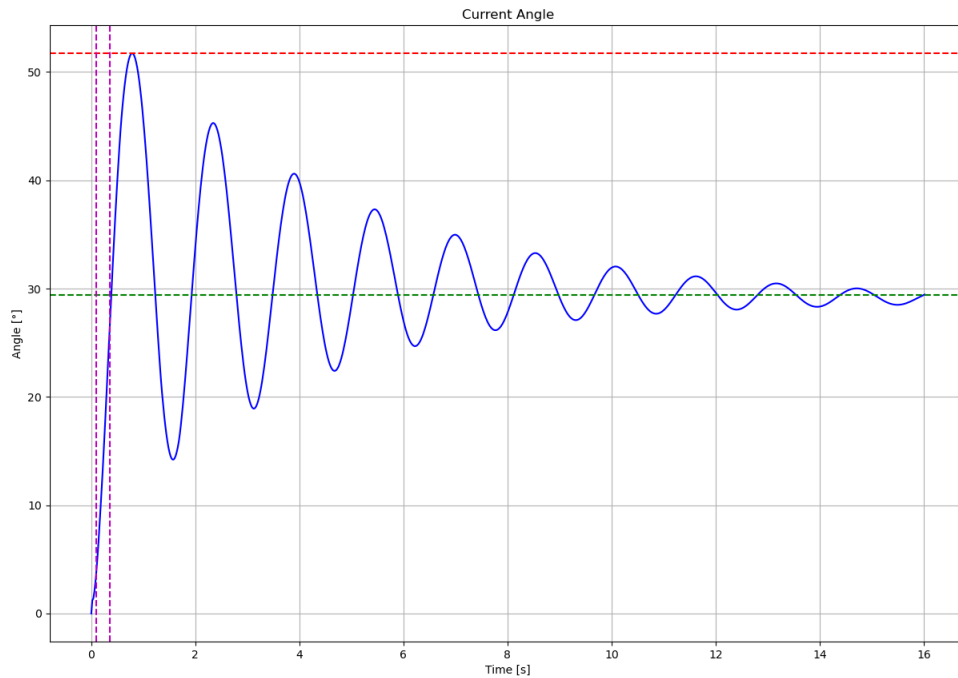


Figura 4.8: Curva de respuesta del modelo PPO ángulo respecto al tiempo.

Capítulo 5

Conclusiones y recomendaciones

5.1. Conclusiones

En el presente documento se demostraron los primeros pasos en la implementación de modelos de aprendizaje reforzado en aplicaciones de control de plantas mediante modelos entrenados con el método *deep Q-networks* (DQN) y optimización de política próxima (PPO).

La selección de los métodos de RL utilizados, respondió directamente a las necesidades expuestas en la matriz de Pugh de la tabla 3.1. La implementación de cada método requirió ajustes, como la necesidad de la discretización del espacio de acciones a 10 posibles en DQN o los cambios efectuados al proceso de entrenamiento base de cada modelo, especialmente al método PPO.

Con base en los resultados expuestos en el capítulo 4, se debe resaltar el papel del diseño de la función de recompensa para lograr resultados satisfactorios con algoritmos de aprendizaje reforzado. Invertir tiempo y esfuerzo en diseñar funciones de recompensa informativas y equilibradas puede mejorar significativamente la eficiencia del entrenamiento, el rendimiento y la eficacia general de los agentes.

En el presente documento los procesos de control del modelo DQN y PPO muestran un comportamiento subamortiguado, lo cual resalta la necesidad de ajustes de la función de recompensa para llegar al ángulo objetivo solicitado o inclusive un porcentaje de error de estado estacionario por debajo del 10 %.

El trabajo futuro podría explorar técnicas más sofisticadas de modelado de recompensas e investigar cómo el diseño de la función de recompensa interactúa con diferentes configuraciones de hiperparámetros para los algoritmos DQN y PPO.

5.2. Recomendaciones

Se recomienda tomar la base expuesta en el presente proyecto y explorar el uso de otros métodos como el DDPG y el SAC, que cuentan con contenido y registro de ser capaces de controlar la planta en cuestión. Por consiguiente, es necesario el acceso a un equipo computacional con características o recursos suficientes para la implementación de dichos modelos.

Dado que se trata de un proyecto con los primeros pasos en implementación virtual, se recomienda ajustar hiperparámetros en diferentes escalas y el ajuste a la función de recompensa para obtener resultados previos representativos a la hora de probar el modelo con la planta real PAMH y evitar posibles daños a la integridad de la planta.

Bibliografía

- [1] F. Golnaraghi y B. C. Kuo, *Automatic Control Systems*, 9.^a ed. USA: Wiley, 2009.
- [2] R. C. Dorf y R. H. Bishop, *Modern Control Systems*, 13.^a ed. USA: Pearson, 2016.
- [3] S. J. Russell y P. Norvig, *Artificial Intelligence A Modern Approach*, 4.^a ed. USA: Pearson, 2021.
- [4] S. Caro, A. Pott y T. Bruckmann, *Cable-Driven Parallel Robots*. Springer, 2023.
- [5] R. Kuman, A. K. Verma, T. K. Sharma, O. P. Verma y S. Sharma, *Soft Computing Theories and Applications*. Springer, 2022.
- [6] J. A. B. Alfaro, *Diseño e implementación de un sistema que simule el comportamiento dinámico de una planta prototipo de control automático utilizando redes neuronales artificiales*, 2022. dirección: https://repositoriotec.tec.ac.cr/bitstream/handle/2238/14289/TF9465_BIB309451_Jorge_Brenes_Alfaro.pdf?sequence=1&isAllowed=y.
- [7] A. Castaño Hernández, J. P. Moreno Beltrán, J. F. Hernández Pérez y R. Villafuerte Segura, «Diseño y control de un sistema balancín con motor y hélice de bajo costo,» *ICBI*, vol. 5, 2018. DOI: <https://doi.org/10.29057/icbi.v5i10.2937>.
- [8] S. Fallas y O. A. Rojas, *Modelo de planta para péndulo amortiguado a hélice*, 2022.
- [9] N. S. Nise, *Control Systems Engineering*. Wiley, 2010.
- [10] K. Ogata, *Ingeniería de Control Moderna*, 5.^a ed. Madrid: Pearson Educación, 2010.
- [11] S. T. Caravaca, *Redes neuronales multimodelo aplicadas al controlador de sistemas*. Universidad Autónoma de Barcelona, 2010.
- [12] S. L. Brunton y J. N. Kutz, *Data-Driven Science and Engineering*. Cambridge University Press, 2021.
- [13] R. S. Sutton y A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 2015.
- [14] V. Mnih et al., *Playing Atari with Deep Reinforcement Learning*, 2013. arXiv: [1312.5602 \[cs.LG\]](https://arxiv.org/abs/1312.5602).
- [15] M. Gupta, «Deep Q Networks (DQN) explained with examples and codes in Reinforcement Learning,» *Data Science in your pocket*, 2023. dirección: <https://medium.com/data-science-in-your-pocket/deep-q-networks-dqn-explained-with-examples-and-codes-in-reinforcement-learning-928b97efa792>.

- [16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford y O. Klimov, *Proximal Policy Optimization Algorithms*, 2017. arXiv: [1707.06347](https://arxiv.org/abs/1707.06347) [cs.LG].
- [17] G. Vadali, *Advantage function in Deep Reinforcement learning*, 2019. dirección: <https://medium.com/mindboard/advantage-function-in-deep-reinforcement-learning-39a0d809c1ff>.
- [18] A. Paszke y M. Towers, *Reinforcement Learning (DQN) Tutorial*, 2024. dirección: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html.
- [19] B. McGill. «Can Q-learning be used for continuous (state or action) spaces?» Mensaje en el foro en línea. (2019), dirección: <https://ai.stackexchange.com/questions/12255/can-q-learning-be-used-for-continuous-state-or-action-spaces>.
- [20] E. Y. Yu, «Coding PPO from Scratch with PyTorch (Part 1/4),» *Medium*, 2020. dirección: <https://medium.com/@eyyu/coding-ppo-from-scratch-with-pytorch-part-1-4-613dfc1b14c8>.
- [21] *The 37 Implementation Details of Proximal Policy Optimization*, 2022. dirección: <https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/>.
- [22] T. P. Lillicrap et al., *Continuous control with deep reinforcement learning*, 2019. arXiv: [1509.02971](https://arxiv.org/abs/1509.02971) [cs.LG].
- [23] T. Haarnoja, A. Zhou, P. Abbeel y S. Levine, *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*, 2018. arXiv: [1801.01290](https://arxiv.org/abs/1801.01290) [cs.LG].
- [24] M. Towers et al., *Gymnasium*, mar. de 2023. DOI: [10.5281/zenodo.8127026](https://doi.org/10.5281/zenodo.8127026). dirección: <https://zenodo.org/record/8127025> (visitado 08-07-2023).
- [25] Y. Hu et al., *Learning to Utilize Shaping Rewards: A New Approach of Reward Shaping*, 2020. arXiv: [2011.02669](https://arxiv.org/abs/2011.02669) [cs.LG].
- [26] E. Y. Yu, *PPO for Beginners*, <https://github.com/ericyangyu/PPO-for-Beginners/tree/master>, 2022.

Apéndice A

Algoritmo del método DQN

Algoritmo 2 Deep Q-Learning con repetición de experiencias. Fuente: [14].

```
1: Inicialice repositorio de experiencia  $D$  con capacidad  $N$ .
2: Inicialice función acción-valor  $Q$  con pesos aleatorios  $\underline{\theta}$ .
3: Inicialice función acción-valor objetivo  $\hat{Q}$  con pesos  $\underline{\theta}^- = \underline{\theta}$ .
4: for  $episodio = 1, \dots, M$  do
5:   Inicialice secuencia  $s_1 = \{x_1\}$  y su mapeo  $\phi_1 = \phi(s_1)$ .
6:   for  $t = 1, \dots, T$  do
7:     if  $\text{rand}_{[0,1]} < \varepsilon$  then
8:        $a_t = \text{acción aleatoria}$ 
9:     else
10:       $a_t = \arg \max_a Q(\phi(s_t), a; \underline{\theta})$ 
11:    end if
12:    Ejecute  $a_t$  y observe  $R_t = R(s_t, a_t, s_{t+1})$  y nuevo valor  $x_{t+1}$ .
13:    Haga  $s_{t+1} = s_t, a_t, x_{t+1}$ , procurese  $\phi_{t+1} = \phi(s_{t+1})$ .
14:    Almacene transición  $(\phi_t, a_t, R_t, \phi_{t+1})$  en  $D$ .
15:    Muestre aleatoriamente  $D$  con un mini-lote.
16:    Asigne
      
$$y_j = \begin{cases} R_j & \text{si en } j+1 \text{ termina episodio} \\ R_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \underline{\theta}^-) & \text{para el resto} \end{cases}$$

17:    Haga un paso de descenso de gradiente sobre  $(y_j - Q(\phi_j, a_j; \underline{\theta}))^2$  con respecto
      a  $\underline{\theta}$ .
18:    Cada  $C$  pasos restablezca  $\hat{Q} = Q$ .
19:  end for
20: end for
```

Apéndice B

Algoritmo del método PPO

Algoritmo 3 Optimización de política próxima (*PPO-Clip*). Fuente: [16].

- 1: Inicialice parámetros de la política θ_0 .
- 2: Inicialice los parámetros de la función de valor ϕ_0 .
- 3: **for** $k = 0, 1, 2, \dots$ **do**
- 4: Recoge el conjunto de trayectoria $D_k = \{\tau_i\}$ al correr la política $\pi_k = \pi(\theta_k)$ en el entorno.
- 5: Calcula el *reward-to-go* \hat{R}_t .
- 6: Calcula la estimación de ventaja \hat{A}_t (usando cualquier método de estimación de ventaja) basada en el actual valor de la función V_{ϕ_k} .
- 7: Actualizar la política al maximizar el objetivo del PPO- Clip:

$$g(\varepsilon, A) = \begin{cases} (1 + \varepsilon)A & A \geq 0 \\ (1 - \varepsilon)A & A < 0 \end{cases}$$

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\varepsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right)$$

típicamente mediante ascenso en gradiente estocástico con Adam.

- 8: Ajustar la función de valor por regresión sobre el error cuadrático medio (MSE):

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2$$

típicamente mediante algún descenso de gradiente.

- 9: **end for**
-