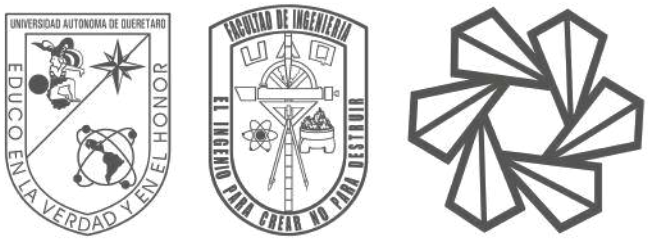
UNIVERSIDAD AUTÓNOMA DE QUERÉTARO



DIRECCIÓN DE INVESTIGACIÓN Y POSGRADO
FACULTAD DE INGENIERÍA

Tesis

Aprendizaje por Refuerzo Aplicado a Control de Posición de una Pata de 3 Grados de Libertad

---

Que como parte de los requisitos para obtener el grado de maestro en ciencias en inteligencia artificial

Presenta:
Ing. Aldo Cervantes Marquez
Asesor:
Dr. Efrén Gorrostieta Hurtado

Sinodales

Dr. Efrén Gorrostieta Hurtado
Presidente
Dr. Juan Manuel Ramos Arreguín
Secretario
Dr. András Takács
Vocal
Dr. Jesús Carlos Pedraza Ortega
Suplente
Dr. Saúl Tovar Arriaga
Suplente

Centro Universitario, Querétaro, México.
Noviembre 2023

Aprendizaje por refuerzo aplicado a control de posición de una pata de 3 grados de libertad

por

Aldo Cervantes Marquez

Dedicatoria

-A mi familia, amigos y personas especiales que me acompañaron durante esta etapa de mi vida y que aportaron su amor incondicional. Siendo este trabajo, el fruto de todo su apoyo.-

Agradecimientos

Agradezco al Consejo Nacional de Humanidades Ciencias y Tecnologías (Conahcyt) por recibir la beca otorgada durante mi estancia en la maestría, a la Universidad Autónoma de Querétaro por ofrecerme la oportunidad de estudiar un posgrado. Al coordinador de mi posgrado, Dr. Saúl, quien siempre apoyó los intereses de los estudiantes de posgrado. A mi asesor, Dr. Efrén, de quien siempre tuve su apoyo y confianza para realizar este trabajo, asi como tambien a todo mi sínodo, quienes me guiaron en la realización de este trabajo y aportaron su conocimiento para tener la mejor calidad en este trabajo. Finalmente quiero agradecer a la facultad de ingeniería, especialmente a su personal academico y administrativo, quienes siempre mostraron profesionalismo y amabilidad durante toda mi estancia.

Índice

Índice de figuras

Índice de tablas

Índice de Algoritmos

1. Resumen

El presente trabajo de tesis presenta el desarrollo e implementación de un algoritmo basado en aprendizaje por refuerzo como controlador en un sistema de control, con la finalidad de realizar trayectorias definidas en 3 ejes (grados de libertad) de una pata robótica que pertenece a un robot hexápodo. Se propone la creación y desarrollo del ambiente basado en el modelo dinámico del sistema electromecánico, cálculo y aproximación de las trayectorias adecuadas y un análisis de los datos obtenidos mediante métricas de evaluación para medir el desempeño del algoritmo ante el sistema. Todo esto con el fin de observar principalmente las capacidades de cómputo del algoritmo, el comportamiento del sistema a lo largo del tiempo (observación de la toma de decisiones) y la viabilidad para trabajos futuros que permitan desarrollar esta línea de investigación para el desarrollo de controladores más robustos, con mayor precisión y que puedan ser aplicados a sistemas cada vez más complejos.

- **Palabras Clave:** Algoritmo de aprendizaje por refuerzo, Sistemas de control, Búsqueda de política, Aprendizaje Automático, Pata de hexápodo.

2. Abstract

The present thesis work introduces the development and implementation of a reinforcement learning-based algorithm as a controller in a control system. The objective is to execute predefined trajectories in three axes (degrees of freedom) of a robotic leg belonging to a hexapod robot. The proposal includes creating and developing the environment based on the dynamic model of the electromechanical system, calculating and approximating suitable trajectories, and analyzing the data obtained using evaluation metrics to measure the algorithm's performance against the system. All of this is aimed at observing primarily the computational capabilities of the algorithm, the system's behavior over time (decision-making observation), and the feasibility for future work that can further develop this line of research for the creation of more robust controllers, with greater precision, applicable to increasingly complex systems.

- **Keywords:** Reinforcement Learning Algorithm, Control Systems, Policy search, Machine Learning, Hexapod Robot Leg.

3. Introducción

El diseño de los mecanismos y actuadores cada día se vuelven más complejos para poder satisfacer las necesidades actuales, por lo que el uso de técnicas de control de sistemas electromecánicos en la resolución de la dinámica de los sistemas se ha convertido en una prioridad para la resolución de tareas más complejas y permite el desarrollo de nuevos elementos mecánicos y electrónicos [1]. Dentro de los métodos para controlar un sistema de control, se encuentran los basados en inteligencia artificial, donde los algoritmos de aprendizaje por refuerzo (RL) destacan en la resolución de problemas en ambientes multivariables a lo largo del tiempo (acción). En donde se pueden modelar ambientes a partir de sus funciones de transferencia y modificar su nivel de abstracción (nivel de granularidad) para adaptar los algoritmos de aprendizaje por refuerzo en diferentes situaciones dentro del mismo ambiente y tenga una interacción correcta [2, 3].

Anteriormente este tipo de algoritmos han tenido problemas de computo debido a la complejidad del algoritmo, sin embargo, actualmente con la innovación y avance tecnológico en las ramas de la informática, electrónica e instrumentación, ha sido posible implementar este tipo de algoritmos en dispositivos cada vez más pequeños y sencillos.

Por otro lado, la aproximación de los modelos matemáticos para representar las funciones de transferencia de los mecanismos, se han generalizado a ecuaciones cuyos elementos componen los principales comportamientos físicos de interés para caracterizar el modelo hasta un nivel de complejidad deseado [4].

Los robots hexápodos tienen una cantidad grande de clasificaciones. Sin embargo, estos generalmente poseen movimientos muy parecidos en sus patas,

por lo que se pueden generalizar las trayectorias que estos generan para realizar una locomoción exitosa [5]. Refiriéndose a los robots hexápodos de 3 grados de libertad con patas electromecánicas, se considera un motor de corriente directa con escobillas y una estructura de de [6, 7]

Por lo que este trabajo de tesis abordará una solución al control de posición de la pata robótica de un hexápodo a partir de trayectorias definidas, observando y visualizando a largo plazo el trayecto del mecanismo. Construyendo un ambiente que caracterice al sistema y tenga un nivel de complejidad granular adecuado para el modelo de aprendizaje por refuerzo, aplicando diferentes algoritmos basados en el aprendizaje por iteración de programación dinámica y aprendizaje Q.

3.1. Problema de Investigación

Los algoritmos basados en aprendizaje por refuerzo tienen relativamente poco tiempo de haber sido implementados en múltiples aplicaciones. Por otro lado, se tiene un sistema de control que se caracteriza por tener no linealidades, por lo que el problema de investigación se abordará desde 3 principales problemáticas (algorítmica, análisis de datos y mecanismo).

Desde un punto de vista algorítmico, la cantidad de variantes que existen de este algoritmo, hiperparametros, capacidad de computo, etc. Se deberán realizar tareas de análisis e implementación de métodos apropiados para el ambiente que se desea controlar [8]. Desde un punto de vista de análisis de datos, se deben determinar las variables que mejor representen al sistema mecánico de una manera simplificada para generar al ambiente y caracterizar sus variables de entrada y salida para poder acoplarlo al algoritmo de aprendizaje por refuerzo [9, 10]. Desde un punto de vista mecánico, los sistemas

articulados con 3 grados de libertad tienen la característica de tener modelos dinámicos no lineales y se pueden presentar escenarios que podrían dificultar la capacidad de movimiento y fallas o variaciones en el sistema eléctrico [11, 12].

Esto implica que la capacidad de movimiento y de configuraciones para hacer trayectorias en un lapso será variable, es decir, la pata tendrá que moverse con distintas fuerzas y modos para poder realizar su siguiente paso [13].

3.2. Justificación

Los algoritmos de aprendizaje basados en inteligencia artificial (específicamente los basados en aprendizaje por refuerzo), han sido ampliamente ocupados para la solución de problemas multi-propósito [14]. Una de sus aplicaciones es dentro del área de control, donde el algoritmo tiende a tomar decisiones de sistemas complejos con no linealidades y varias entradas y salidas. Todo esto ha desencadenado la creación de nuevos modelos, diseños y retos para poder controlar sistemas más elaborados, ya sea de tipo mecánico, eléctrico, físico, etc. Por lo que es de mucho interés el estudio de estas tecnologías y métodos para poder conocer su comportamiento y desempeño ante sistemas de alta variación de posibilidades de salida ante las mismas entradas, esto permitirá en el caso especifico de la robótica y el control, un desarrollo en la exploración y experimentación de sistemas de robots que puedan tener movimientos más naturales y acordes a los de un animal o insecto [15].

Por lo que una vez dicho esto, se observa como una opción viable el uso de aprendizaje por refuerzo en este sistema debido a su alta complejidad y no linealidad del mismo.

3.3. Antecedentes

El uso de los algoritmos de aprendizaje por refuerzo (abreviados como RL) se basan en los estudios y aplicación de las ciencias básicas y de ingeniería como el de la informática, psicología, estadística y ciencias computacionales. El uso del comportamiento y aprendizaje humano han permitido a que el algoritmo cobre importancia puesto que se basa en la psicología conductal, en la que se tiene al algoritmo actuando sobre un ambiente determinado (más no necesariamente conocido), un tercero (el cual también es albergado dentro del algoritmo y suele ser otro algoritmo embebido) juzga su comportamiento y con base en sus resultados, puede castigarlo o recompensarlo, con el fin de que aprenda mediante la experiencia propia y la retroalimentación recibida [16].

En los últimos años ha tomado fuerza este algoritmo debido a su capacidad de aprender de una manera autónoma en cada momento y con posibilidad de manejar múltiples variables tanto de entrada como de salida. Además de no poseer las limitaciones del aprendizaje supervisado en el que es forzoso conocer a profundidad el sistema para entrenarlo de una manera adecuada, ni tampoco cuenta con las limitaciones del aprendizaje no supervisado en el que no se tienen siempre herramientas necesarias para un aprendizaje veloz o ágil. Sin embargo, toma sus características como base de su aprendizaje (véase Figura 1) [17].

También tiene la capacidad de no solo predecir o clasificar objetos. Sino también, tomas de decisiones más complejas dando lugar a entornos multivariables, como por ejemplo un sistema de visión artificial aplicada a un automóvil autónomo, en donde no solo será necesario catalogar objetos como señalamientos, personas, otros carros, etc. Por lo que deberá controlar

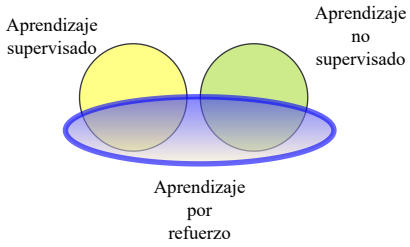


Figura 1: Diagrama de aprendizaje por refuerzo [17].

también los movimientos del carro y comprender las situaciones a las que se encuentre, teniendo la noción de las condiciones de manejo adecuadas para tener un trayecto seguro a lo largo del viaje [17, 18].

El algoritmo de aprendizaje por refuerzo tiene su primer antecedente en 1972 con una investigación que se enfoca en ideas primordiales para el uso e implementación del algoritmo en sistemas adaptativos basados en inteligencia artificial, abriendo una brecha en la investigación del aprendizaje por refuerzo mediante el uso de sistemas de aprendizaje hedonísticos [19].

Posteriormente, entre los años 1989 y 1998 se retomó esta técnica de inteligencia artificial debido a que las capacidades de computo de ese año no eran las suficientes para poder correr el algoritmo. En 1989 se realizó un algoritmo de aprendizaje por refuerzo basado en Q-Learning, una técnica que es la más popular actualmente, que hace uso de la programación estocástica, y se obtuvo un método para resolver un problema para optimizar problemas de comportamiento de animales según el ambiente en el que se desenvuelven [20].

Entre esos años, en 1992, se desarrolló un trabajo complementario al realizado en 1989, proponiendo el algoritmo basado en aprendizaje por refuerzo profundo, donde el agente aprende de manera óptima en un dominio Markoviano mediante el uso de programación dinámica [21]

En 1998 se escribió un libro que comprendía con todas las reglas formales y ejemplos explícitos con aplicaciones en el campo de la inteligencia artificial y la ingeniería [22]. Este libro marcó un hito en la formalización de este algoritmo y sus reglas e implicaciones.

Estos algoritmos fueron poco investigados en esos años, generando pocas variantes del mismo, pero en los últimos 10 años se desarrollaron muchos trabajos del aprendizaje por refuerzo. Sobresale el aprendizaje por refuerzo profundo (DRL). El cual es basado en redes neuronales combinadas con el algoritmo RL. Destacan los casos de investigación AlphaGo (así se le llamó al programa que aprendió a jugar uno de los juegos más antiguos y famosos del mundo, Go) quien jugó contra uno de los mejores jugadores de ese momento (Lee Sedol) [23, 24]. El juego, si bien no aportó conceptos nuevos a los que en esos momentos se conocían, se consideró como una aportación a la implementación, integración y capacidades de aprendizaje del algoritmo (véase Figura 2).

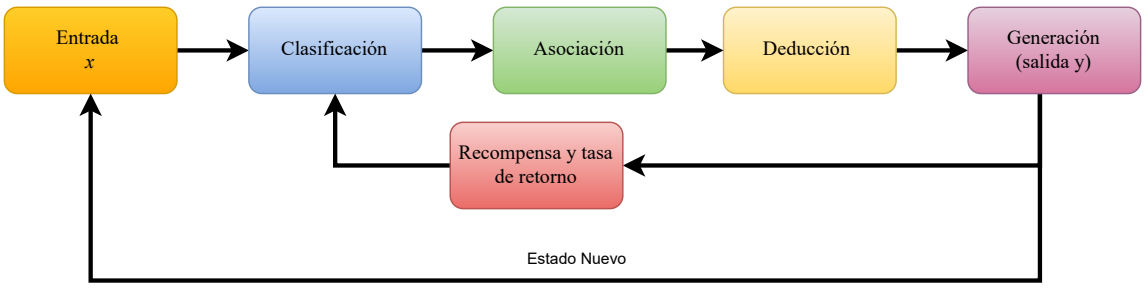


Figura 2: Algoritmo de AlphaGo basada en redes neuronales profundas para la toma de decisiones [24].

Después de ese momento, la cantidad de trabajos relacionadas con DRL fueron incrementados, así como también las aplicaciones fueron diversificadas hasta los sistemas de control que son los de mayor interés para esta investigación. A continuación se muestran trabajos de importancia para el desarrollo del RL. En 2018 se desarrolló un estudio sobre las capacidades del algoritmo

de RL basadas en experiencias pasadas, con el fin de obtener las aplicaciones y parámetros más adecuados según la aplicación (se enfocan en aplicaciones físicas) en la que el agente se encuentre sometido [25]. Se obtuvieron algunos factores de interés para el uso del algoritmo para responder a las siguientes preguntas:

- ¿Cómo elegir la capacidad adecuada para que la experiencia obtenida sea de utilidad?
- ¿Cuáles experiencias son más enriquecedoras para el aprendizaje del sistema?
- ¿Cómo muestrear u obtener las experiencias más importantes?

Todo esto con la finalidad de que el algoritmo pueda realizar sus aprendizaje de una manera eficiente, veloz y estable (que aprenda, retenga, y mejore su aprendizaje).

En 2020 Nguyen et al. publicaron un artículo en donde se muestra un Algoritmo de DRL multi-objetivo (MODRL) [26], el cual consta de un algoritmo que combina las estrategias propias de RL single-policty y multi-policity para la resolución de problemas multi-propósito basada en Q-Networks. Se logró proponer un MODRL escalable de alto desempeño, implementado en lenguaje en Python, marcando un futuro en la implementación y desarrollo de estos algoritmos basados en redes neuronales profundas.

En 2019 Dornheim et al. desarrollaron un algoritmo de RL aplicado a procesos de manufactura en procesos repetitivos como son las prensas hidráulicas, las cuales prensan sobre moldes figuras especificas. Teniendo un agente de control óptimo para conocer la posición adecuada mediante un modelo de predicción. Propusieron un modelo libre con la implementación de

Q-Learning, permitiendo la adaptabilidad de los procesos al variar continuamente las condiciones a través del aprendizaje [27].

En 2015 Polydoros et al. publicaron un articulo, el cual desarrolla un algoritmo de DRL para controlar un robot manipulador mediante un sistema de control alternativo de aprendizaje autónomo. El algoritmo pretende no identificar el sistema, sino que el algoritmo realice una inspección y aprenda como controlarlo mediante dinámica inversa e inferencia Bayesiana. Como resultado se obtuvo un algoritmo que se puede adaptar a los cambios en tiempo real de la dinámica inversa tomando en cuenta la frecuencia de muestreo, redes neuronales auto-organizadas y el depósito computacional del algoritmo [28].

En 2019 Arranz, et al. realizaron una tesis en donde se propone un algoritmo de DRL aplicado al juego CartPole (un juego clásico para el aprendizaje autónomo) el cual consiste en un carro que contiene una vara acoplada con un eje al carro, y debe ser equilibrada, algo análogo a un péndulo invertido montado sobre un carro (véase Figura 3), fue implementado en Python, caracterizando y discretizando los casos con los que se puede controlar el sistema sin necesidad del la función de transferencia del sistema o los parámetros que lo componen, dividiendo en 2 estados de acción.

- Si la vara se mueve hacia la izquierda, mover el carro a la izquierda.
- Si la vara se mueve hacia la derecha, mover el carro a la derecha.

Como resultado se obtuvo una buena respuesta del sistema. Sin embargo, fue necesario ajustar la recompensa adecuada en las iteraciones requeridas, se supuso que no hay un tiempo muerto por muestreo ni por procesamiento por lo que se idealizó el sistema [29].

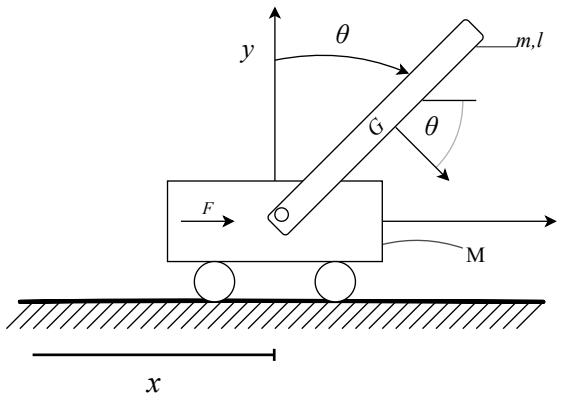


Figura 3: Sistema de péndulo invertido sobre carro (CartPole) [29].

En 2021 se publicó un trabajo en donde se realizó un algoritmo basado en algoritmos genéticos que pueda gestionar al agente del programa de RL, mejorando un las tareas multi-agente aplicada a la industria de procesos en China, proponiendo un incremento a la autonomía de los sistemas basadas en tecnologías de la industria 4.0, obteniendo como resultado un modelo híbrido con un mejor rendimiento que el RL aislado [30].

En el año 2020 se publicó un trabajo en donde se muestra un sistema que controla el desfogue de pequeñas presas para circular el agua por donde sea requerido, mediante un sistema en tiempo real y redes de sensores para obtener datos de entrada. Se obtuvo como resultado un sistema que al sintonizar las ganancias de la recompensa, las redes neuronales profundas trabajarán de una manera más sensible para controlarla cantidad de agua en varios puntos (lagos o pequeñas presas) al mismo tiempo [31].

3.4. Estado del Arte

En la Tabla 1 se muestra, de una manera condensada y complementaria, un compilado de los trabajos de mayor interés y que han sido desarrollados

en los últimos 5 años, siendo los que aportan mayor utilidad al contexto del trabajo de investigación, extrayendo la información más importante (aportación y desventajas) como un punto de partida para el desarrollo y áreas de oportunidad a tratar.

Tabla 1: Estado del arte.

Ref.	Fecha	Titulo	Aportación	Desventaja
[25]	2018	Experience Selection in Deep Reinforcement Learning for Control	Seleccionaron algunas aplicaciones donde el algoritmo se puede desenvolver mejor.	Muchas limitantes en cuanto a control posiblemente mal manejadas (frecuencias de muestreo, ruido del sensor, limitaciones de computo).
[31]	2020	Deep reinforcement learning for the real time control of stormwater systems	Sistema de control de presas con redes multi-sensoriales y puede tomar acciones según la duración y cantidad de lluvia en la época del año que se encuentre.	Difícil obtención de ajuste de recompensa y arquitectura del sistema, se suele realizar a prueba y error para hacer ajustes.
[32]	2020	Reinforcement Learning Compensation based PD Control for Inverted Pendulum	Q-Learning + Control PD genera un controlador robusto y con buena respuesta ante cambios de parámetros.	Q-Learning solo, no se estabiliza rápidamente y es muy fluctuante.
[33]	2019	Reinforcement Learning Compensation based PD Control for a Double Inverted Pendulum	Q-Learning + Control PD trabajan mejor juntos que separados.	Difícil obtención de la recompensa adecuada. 1hr para que el algoritmo pueda aprender (realiza más de 200,000 iteraciones).
[27]	2020	Model-free Adaptive Optimal Control of Episodic Fixed-horizon Manufacturing Processes Using Reinforcement Learning	Integración de control adaptativo con aprendizaje por refuerzo. Suposición del sistema de control. Preparado para cambios de parámetros.	Gran cantidad de iteraciones. Uso de neuronas auxiliares.

[34]	2021	A Probabilistic Interpretation of Self-Paced Learning with Applications to Reinforcement Learning	Uso de curricula para entrenar evita atascarse en valores locales. Uso de algoritmos de optimización como Underlying Optimization.	Alta capacidad recurso de computo (66 neuronas). Muchos Steps (5000).
[35]	2021	Redes neuronales y aprendizaje por refuerzo en el control de turbinas eólicas	Uso de neuro-estimadores para controlar sistemas no lineales con apoyo de controladores tradicionales.	Solo funciona en turbinas de baja potencia y de configuraciones sencillas, por lo que el algoritmo no es tan robusto.
[26]	2020	A multi-objective deep reinforcement learning framework	RL aplicado a multi-objetivo. Bases muy sólidas para trabajos futuros	Limitaciones en funcionamiento multi-policy y tiempo de procesamiento.
[36]	2021	Adaptive Supply Chain: Demand–Supply Synchronization Using Deep Reinforcement Learning	Uso de optimización por Proximal Policy, aplicado a cadenas de suministro e inventarios y comparada con otros métodos de optimización.	Aprendizaje lento y con fallas debido a la suposición de la visibilidad end-to-end, por lo que para casos reales podría no funcionar del todo bien.
[37]	2021	Reward boosts Reinforcement-based motor Learning	Enfatiza la importancia de la obtención de la recompensa adecuada para el algoritmo con el fin de mejorar su aprendizaje, mediante un método de realimentación y recompensa, todo esto, mientras continua aprendiendo.	El RL por sí solo, no permite ni mejora la influencia para el motor de aprendizaje.
[38]	2021	Adaptive workload adjustment for cyber-physical systems using deep reinforcement learning	Uso de *Q-Learning* para auxiliar CPS (cyber-physical systems) proponiendo una reducción de costos en estos módulos, tolerante a fallas de los mismos.	Problemas de implementación si no se hace de manera correcta, difícil de implementar, puede haber problemas contraproducentes.
[39]	2021	Deep reinforcement learning for multi-contact motion planning of hexapod robots	Uso de aprendizaje por refuerzo profundo en ambientes variados, resuelve problemas de locomoción.	Mucho tiempo de entrenamiento.

[40]	2022	Fuzzy double deep *Q-network-based* gait pattern controller for humanoid robots	Uso de una doble red neuronal con lógica difusa como alternativa.	Sistema demasiado complejo para la propuesta, sin embargo, se logró lo propuesto.

Se observa que los trabajos relacionados se enfocan en sistemas de caja negra o sistemas cinemáticos, por lo que no se contemplan las suficientes variables que describan al modelo. Adicionalmente, los trabajos muestran un tiempo de entrenamiento alto basado en una gran cantidad de épocas y una estructura de algoritmo complejo debido a la poca observabilidad de los sistemas profundos, lo que dificulta su implementación en dispositivos de recursos limitados. Los modelos del estado del arte controlan únicamente una variable y esto limita su alcance.

4. Fundamentación Teórica

4.1. Aprendizaje por Refuerzo

Para conocer con mayor detalle el aprendizaje por refuerzo es necesario conocer el concepto de *Machine Learning* (aprendizaje automático en español), el cual se puede definir como un subcampo de la inteligencia artificial, el cual se caracteriza por la capacidad de los algoritmos de poder resolver problemas sin necesidad de que se les programe con todas las reglas requeridas, prediciendo casos particulares [41]. Aprendiendo u observando patrones definidos o tendencias en los datos que procesan, construyendo un modelo que permita describir esos comportamientos [17]. A partir de esto es que se puede definir al aprendizaje por refuerzo como una parte del aprendizaje automático.

El aprendizaje por refuerzo consiste en un modelo o espacio de trabajo (ambiente) en el que un agente deberá explorar las posibilidades del espacio y mediante una recompensa o penalización que es dada según sus resultados. La idea principal detrás del algoritmo es el de maximizar la recompensa (sus intentos satisfactorios) a lo largo de un periodo de tiempo y de aprendizaje, creando una estrategia para lograrlo [22, 42]. De una manera simplificada puede ser visto el algoritmo de aprendizaje por refuerzo como se muestra en la Figura 4.

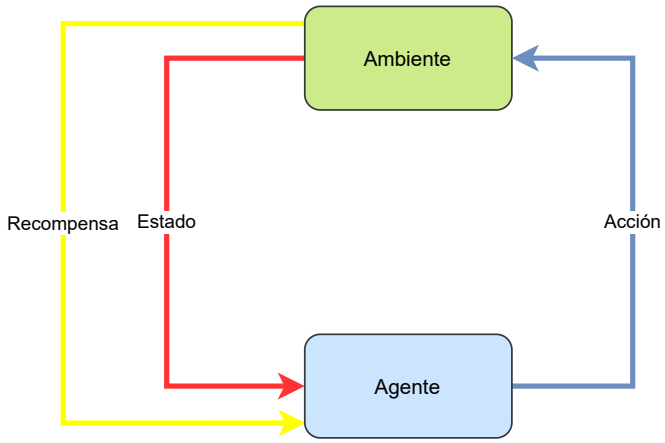


Figura 4: Modelo de bloques simplificado del Aprendizaje por Refuerzo [22].

Desde un punto de vista algorítmico e iterativo, es posible definirlo como se muestra en el Algoritmo 1 [43].

Algoritmo 1 Proceso iterativo del aprendizaje por refuerzo.

```
Inicializar estado
    Repetir (Para cada paso del episodio):
        Seleccionar estado
        Desarrollar estado; observar Recompensa y proximo.estado
        estado ← proximo.estado
```

El aprendizaje por refuerzo puede ser visto desde un punto de vista matemático como la resolución de un proceso de decisión de Markov (MDP)

[43, 44], el cual es representado como se muestra en la Ecuación (1).

$$P_{k+1} = (S_{k+1}|S,a) \tag{1}$$

Por lo que se puede explicar la ecuación en que existe una función de transición P para el estado S_{k+1} (es decir, el siguiente estado), basado en el estado anterior S y una acción realizada en el mismo estado a.

Esto implica que la acción dependerá y se generará a partir de una política π que posee parámetros de entrada θ en un estado actual S_t como se muestra en la Ecuación (2).

$$a = \pi_\theta(S_t) \tag{2}$$

Este concepto puede ser representado como una maquina de estados con una probabilidad de transición P a partir de las acciones a (véase Figura 5).

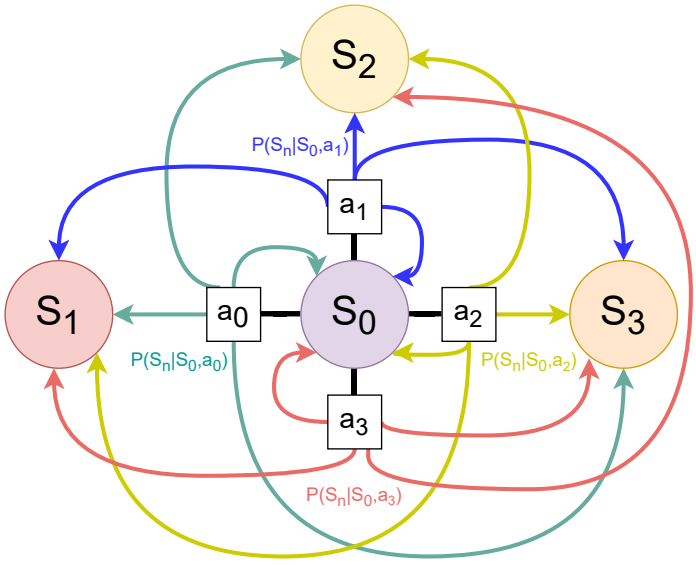


Figura 5: Principio de decisión de Markov en aprendizaje por refuerzo [10].

Matemáticamente es común representarlo en matrices para poder manejar todos los estados de una manera ordenada y relacionada entre si [45]. Teniendo una matriz de transición M_t, en donde los renglones deben sumar una probabilidad de $\sum_m p(n,m) = 1$,representada:

$$m_t = \begin{bmatrix} p(1,1) & p(1,2) & p(1,3) & \cdots & p(1,m) \\ p(2,1) & p(2,2) & p(2,3) & \cdots & p(1,m) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ p(n,1) & p(n,2) & p(n,3) & \cdots & p(n,m) \end{bmatrix} \tag{3}$$

donde las probabilidades se calculan mediante la frecuencia de la combinación de las clases:

$$p(n,m) = \frac{f(n,m)}{\sum f(m_t(:,m))} \tag{4}$$

Por otro lado se tiene que la matriz de probabilidad de estado con la forma:

$$s_t = \begin{bmatrix} p(s_1) & p(s_2) & p(s_3) & \cdots & p(s_n) \end{bmatrix} \tag{5}$$

Donde finalmente se obtiene su valor nuevo de probabilidad de estado s_{t+1}:

$$s_{t+1} = s_t \times m_t \tag{6}$$

Una vez mostrado el proceso de funcionamiento del algoritmo en función de entradas y salidas, a continuación se muestran los elementos y conceptos más importantes sobre el aprendizaje por refuerzo [46, 41, 47]:

- **Agente:** Es el elemento que únicamente es tomar decisiones por lo que hace las acciones para resolver el problema.

- **Política:** Es la forma en la que el agente puede actuar a un determinado problema. En este caso puede ser una regla de control que el agente puede actuar sobre el sistema. De una manera general se puede ver como una función de la forma $\pi(S,a)$ que asocia el estado (S) y la acción del agente (a). Cabe mencionar que existen 2 tipos de politicas principalmente, las cuales se basan en el conocimiento y accionar del agente los cuales son:

  - **Basado en modelo (model-based)**: donde el agente conoce una descripción completa del entorno.
  - **Sin modelo (model-free)**: donde el agente no conoce la dinámica del entorno.

- **Entorno o ambiente:** Es el modelo sobre el cual el agente opera, respondiendo con estados u observaciones a las acciones del agente, es importante mencionar que existen dos tipos de modelos sobre los que actúa el agente los cuales son:

  - **Determinista**: Es un entorno en el que únicamente hay un estado siguiente posible para una acción dada.
  - **Estocástico**: Es aquel en el que el entorno reacciona de una manera diferente a las mismas acciones del agente

- **Estado:** Se refiere al momento en el que se encuentra el agente, basado en las condiciones variantes del ambiente (algo análogo a las iteraciones de un algoritmo). se suele representar con la letra S y es el elemento que a su salida puede obtener una colección de las variables de interés que sean requeridas.

- **Función de transición:** Como se sabe, a cada acción se pasa a un siguiente estado, sin embargo, la probabilidad de pasar de un estado a

otro, dependerá de una función de transición entre los estados, optando con la de mayor éxito para maximizar la recompensa.

- **Recompensa:** Es una respuesta directa sobre las acciones del agente. Aportando un logro a la tarea que realiza el agente durante cada estado.

- **Episodio:** La tarea que realiza puede tener o no un final natural. Esto significa que el entorno tendrá un estado final (como por ejemplo un juego de ajedrez o la cantidad de gasolina de un automóvil autónomo) se les llaman *Tareas Episódicas*, las tareas que no tienen final definido, se les llaman *Tareas continuas*. La secuencia de pasos (*timesteps*) de principio a fin de una tarea episódica se le llama *episodio*, en el caso de las tareas continuas se les llama *Trayectoria*.

- **Retorno o tasa de descuento:** Es la suma de las recompensas recolectadas durante un episodio, por lo que el agente no conocerá su recompensa sino hasta terminar el episodio. Determina que tan bien aprovecha las primeras iteraciones del episodio para poder obtener la mejor recompensa y de esta manera, el algoritmo tenga un aprendizaje incremental. Se define como:

$$G_t(E) = \sum_{j=0}^{n_t} \gamma^j r_j \tag{7}$$

Todo este proceso puede ser modelado de una manera más completa con los elementos más importantes en un sistema de control como se muestra en la Figura 6.

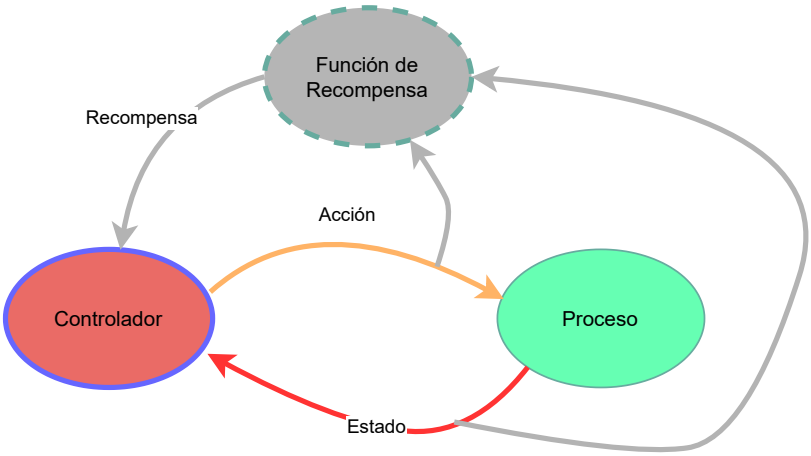


Figura 6: Interacción de un sistema de control con aprendizaje por refuerzo [47].

4.2. Políticas de Aprendizaje por Refuerzo

Debido a la naturaleza del problema, es necesario conocer las posibles políticas que pueden resolver las trayectorias para el sistema dinámico. Por lo que el ambiente puede tomar la forma de ser estocástico o determinista según sea el caso. Esto implica que se puede emplear una amplia gama de algoritmos con los que se pueden trabajar como se muestra en la Figura 7 [48].

Se enfocará en las políticas libres de modelo, puesto que suelen ser más robustas y las más utilizadas según lo revisado en el estado del arte (Tabla 1) [49, 50].

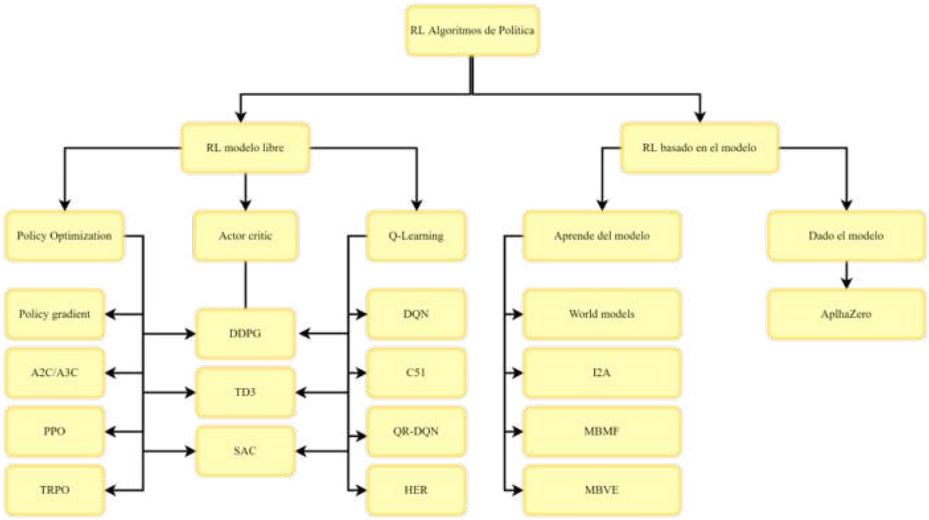


Figura 7: Clasificación de las principales políticas de aprendizaje por refuerzo [48].

4.2.1. Programación Dinámica y Aprendizaje por refuerzo

Este concepto nace a partir de la problemática de la complejidad computacional de la optimización de problemas, en donde se tienen procesos repetitivos y secuenciales [9, 51]. Proponiendo una resolución mediante el uso de dos ideas principales: la primera se enfoca en subdividir el problema en sus componentes secuenciales o temporales con un enfoque Markoviano, su segundo enfoque se basa en el de almacenar los resultados a estos subproblemas para evitar cálculos repetitivos [52, 53].

Por lo que de igual manera que el MDP se dividen estados y acciones para el modelo de aprendizaje por refuerzo, se almacenan los aprendizajes en tablas de consulta para el algoritmo. De este modo es que puede realizar una consulta sobre las acciones tomadas ante situaciones similares (véase Figura 8).

Acción (A) / Estado (S)	A_0	A_1	$\cdots$	A_{n-1}	A_n
S_0	$Q(S_0, A_0)$	$Q(S_0, A_1)$	$\cdots$	$Q(S_0, A_{n-1})$	$Q(S_0, A_n)$
S_1	$Q(S_1, A_0)$	$Q(S_1, A_1)$	$\cdots$	$Q(S_1, A_{n-1})$	$Q(S_1, A_n)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
S_{m-1}	$Q(S_{m-1}, A_0)$	$Q(S_{m-1}, A_1)$	$\cdots$	$Q(S_{m-1}, A_{n-1})$	$Q(S_{m-1}, A_n)$
S_m	$Q(S_m, A_0)$	$Q(S_m, A_1)$	$\cdots$	$Q(S_m, A_{n-1})$	$Q(S_m, A_n)$

Figura 8: Tabla estado/acción para resolución de problemas con programación dinámica (Fuente: Creación propia).

En donde se aprecia que la tabla de estado/acción consta de valores Q que consisten en la información y/o experiencia adquirida en el estado S_t y en la acción A_t en el paso t de la secuencia (episodio) en la que se encuentre.

Para poder aplicar este concepto al aprendizaje por refuerzo, se debe realizar una metodología de tipo *SARSA* el cual por sus siglas (*State-Action-Reward-Sate-Action*), permite hacer un analisis de la secuencia de los estados a los que se está dirigiendo el algoritmo [54]. Por lo que se puede diseñar un algoritmo de aprendizaje por refuerzo basado en estado-acción-transición (tipo *SARSA*) como se muestra en la Figura 9.

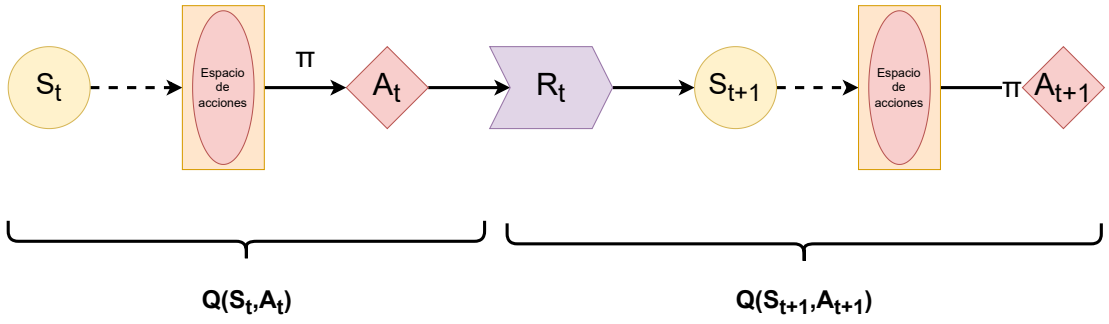


Figura 9: Proceso SARSA en algoritmo de aprendizaje por refuerzo (Fuente: Creación propia).

Teniendo como los principales algoritmos de programación dinámica (políticas) para problemas estocásticos: *Value-iteration*, *Q-Learning*, *SARSA*. Los cuales se caracterizan por tener el mismo principio de funcionamiento en

cuanto a su proceso de estado-acción-transición [55].

La política de *Value-Iteration* se basa en la probabilidad de transición entre los estados posibles $P(s'|s,a)$ según la acción realizada a y la recompensa R con una función de valor $V(s)$ que se obtiene mediante la Ecuación (8) [10].

$$\begin{aligned} Q(s,a) &\leftarrow \textstyle\sum_{s'} P(s'|s,a) \cdot [R(s,a,s') + \gamma \cdot V(s')] \\ V(s) &= \max_a Q(s,a) \end{aligned} \tag{8}$$

En cambio la política *Q-Learning* se basa en la *Ecuación de Bellman* en el que se considera un factor de aprendizaje α que tiene la acción actual, tomando como 1 la probabilidad de llegar a un estado determinado.

$$\begin{aligned} Q(s,a) &\leftarrow (1-\alpha)(Q(s,a)) + \alpha[R(s,a,s') + \gamma \cdot V(s')] \\ V(s) &= \max_a Q(s,a) \end{aligned} \tag{9}$$

Es posible agregar un factor de entropía probabilistica H a este método, en el que se calcula el promedio de la información que es producida por una fuente de datos, permitiendo explorar más regiones no exploradas. Generando una variante de política llamada *Soft Q-Learning* [55].

$$\begin{aligned} H(p) &= -\textstyle\sum_i p_i \log p_i = \mathbb{E}[-\log p_i] \\ Q_{\text{soft}}(s,a) &= R(s,a) + \mathbb{E}_\pi[\textstyle\sum_{i=1}^{\infty} \gamma^i (R(s',a') + H(\pi(\cdot|s'))] \end{aligned} \tag{10}$$

Finalmente la política *SARSA* tiene las mismas consideraciones que *Q-Learning*. Sin embargo, *SARSA* toma en cuenta que su función de valor el valor del siguiente valor Q estado-acción.

$$\begin{gathered} Q(s,a) \leftarrow (1-\alpha)(Q(s,a)) + \alpha[R(s,a,s') + \gamma \cdot V(s')] \\ V(s') = Q(s',a') \end{gathered} \tag{11}$$

Teniendo un componente extra que es la función *Greedy* (ε), la cual permite realizar una decisión entre la exploración y la explotación de estados a través de un umbral probabilístico. Esto significa que el algoritmo tendrá una probabilidad de poder explorar estados diferentes mediante el uso de acciones aleatorias, todo esto con el fin de poder encontrar mejores soluciones a cambio de tener malas recompensas en el camino. Por otro lado la explotación permite realizar las mejores acciones según lo aprendido en la exploración. Esto genera un dilema en el que se tiene que tener un equilibrio entre la ejecución de estas acciones, titulada el dilema de exploración vs explotación [56].

Esta función ε debe ser adecuada para priorizar la generalización del problema y en consiguiente realizar un mejor desempeño mientras se encuentran las soluciones al problema. Esta función es muy adaptable al problema que se está resolviendo, teniendo la particularidad de tener un rango entre $[-\infty, \infty]$ y una imagen entre [0,1]. A continuación en la figura 10 se muestran las funciones ε básicas con un dominio en las épocas.

Permitiendo realizar un algoritmo que combina el proceso de decisión de Markov, la programación dinámica con las tablas de estado/acción, sus algoritmos (políticas deterministas) basadas en aprendizaje por refuerzo y la función *Greedy* en el Algoritmo 2 .

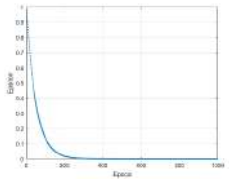
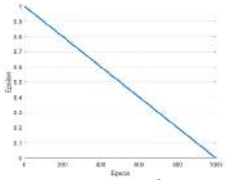
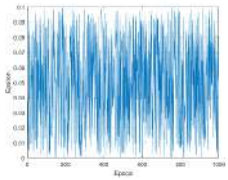
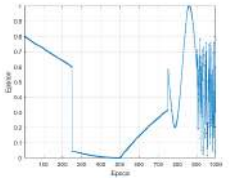
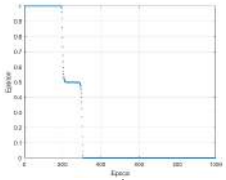
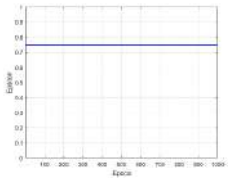
Exponencial	Lineal	Aleatoria
 $\epsilon(e_p) = e^{-ke_p}$	 $\epsilon(e_p) = 1 - \frac{e_p}{e_f}$	 $\epsilon = P(0,1)$
A trozos	**Escalonada**	**Constante**
 $\epsilon = \begin{cases} f_1(\epsilon)\ para\ [\epsilon_0, \epsilon_1] \\ \vdots \\ f_m(\epsilon)\ para\ [\epsilon_{m-1}, \epsilon_m] \end{cases}$	 $\epsilon = \frac{-1}{1 + e^{-\frac{(a*e_p - b)^{\frac{2n+1}{2}}}{c}}} + 1$	 $\epsilon = a, \quad a \in [0,1]$

Figura 10: Principales funciones ε (Fuente: Creación propia).

Algoritmo 2 Programación dinámica en proceso de aprendizaje por refuerzo

Inicio
  Inicializar *V(s)=0* **Para todo** $s \in S$
  Mientras $s \text{ ¡} = s_{final}$:
    Si $f(\epsilon) \geq \Theta$: ## Proceso de exploración
      Actuar $a = rand(Q(s, A))$
      Actualizar $Q(s,a), V(s) = \max Q(s, A)$
    Si $f(\epsilon) < \Theta$: ## Proceso de explotación
      Actuar $a = \max Q(s, A)$
    Regresar Parámetros nuevos $(s \leftarrow s', r, Q(s,a) \leftarrow Q(s', a'))$
Fin

4.3. Pata Robótica de 3 grados de libertad

La pata robótica a modelar será basada en una mecánica natural de un hexápodo, la cual consta de 3 grados de libertad, con el fin de obtener las posiciones adecuadas para los movimientos del robot [57]. Este sistema es considerado no lineal, debido a su control de posición con sus ángulos de

inclinación y la obtención de sus modelos a partir de funciones no lineales.

El sistema consta de 3 ángulos θ_1,θ_2 y θ_3. Estos ángulos tienen su rango de movilidad para el movimiento de la pata del robot mediante eslabones unidos por barras (véase Figura 11).

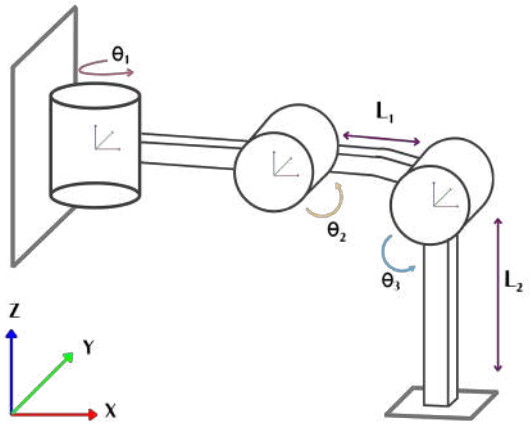


Figura 11: Diagrama de cuerpo libre de la pata robótica con 3 grados de libertad [58, 59].

Estos ángulos siguen trayectorias parabólicas (véase Figura 12) mediante las siguientes ecuaciones [59].

$$\theta_1 = d\phi - A\phi(cos(\zeta) - 1) \tag{12}$$

$$\theta_2 = d\beta - A\beta(cos(\zeta) - 1)e^{k1\zeta} \tag{13}$$

$$\theta_3 = d\chi - A\chi(cos(\zeta) - 1)e^{k1\zeta} \tag{14}$$

donde $d\phi$,$d\beta$,$d\chi$ están dados por los ángulos θ_1,θ_2,θ_3. La variable $A\phi$ define la longitud del paso de la pata, y $A\beta$, $A\chi$ definen la altura del paso y ξ es un factor de movilidad del ángulo (variable independiente en radianes), finalmente $k1$ es la variable de la pendiente al final del paso, esta se puede representrar como una función de tipo gaussiana o sigmoide.

$$k1 = \frac{1}{c^2\sqrt{2\pi}} e^{-\frac{(\zeta - b)^2}{2c^2}} \tag{15}$$

Donde b, c son constantes que dependerán de algunas variables físicas de la pata robótica, por lo que no existe un método exacto para calcularlas [59].

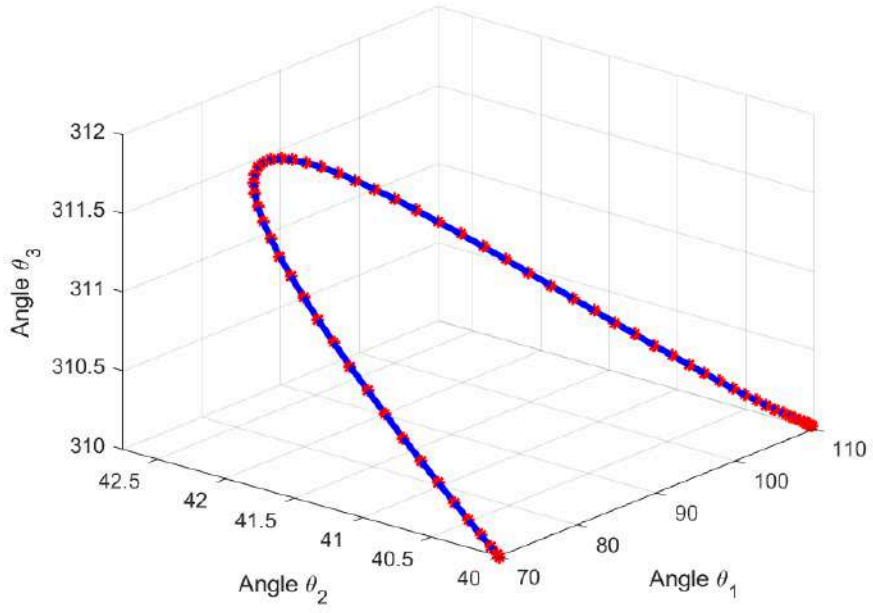


Figura 12: Trayectoria parabólica de la pata robótica en un paso [58, 59].

El modelo dinámico del sistema completo se simplifica mediante una suma de funciones no lineales en la Ecuación (16) [58].

$$D(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) + b(\dot{\theta}) = \tau \tag{16}$$

Donde $D(q)$ es la matriz de inercia, $c(q, \dot{q})$ son los términos de fuerza centrifuga y Coriolis, $g(q)$ es el efecto de la gravedad, $b(\dot{q})$ es el efecto de la fricción y τ es el momento de torque generado. Debido a la cantidad de grados de libertad, se ajustó en un modelo matricial donde cada renglón y columna representa al grado de libertad en cada ángulo, y cada variable a_x, b_x, c_x, f_x representa una función diferente $F_{a,b,c,f}(\theta_1, \theta_2, \theta_3)$, las cuales se pueden observar con detalle en los Anexos.

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ b_{21} & 0 & b_{23} \\ b_{31} & b_{32} & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_2^2 \\ \dot{\theta}_3^2 \end{bmatrix} + \dots$$
$$+ \begin{bmatrix} c_{11} & c_{12} & 0 \\ c_{21} & 0 & c_{23} \\ 0 & 0 & c_{33} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1\dot{\theta}_2 \\ \dot{\theta}_1\dot{\theta}_3 \\ \dot{\theta}_2\dot{\theta}_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & f_{22} & f_{23} \\ 0 & 0 & f_{33} \end{bmatrix} \begin{bmatrix} m_1 g \\ m_2 g \\ m_3 g \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \tag{17}$$

Como actuador se utilizará un modelo de posición de un motor eléctrico con parámetros de entrada de corriente y 2 constantes que se basan en su función de transferencia y que fueron obtenidas a partir de experimentación [60]. Se puede expresar su función de transferencia en transformada de Laplace $\mathscr{L}$ a partir de la siguiente Ecuación.

$$\frac{\theta(s)}{I(s)} = \frac{K}{s(s+a)} \tag{18}$$

Aplicando el teorema del corrimiento, es posible obtener su ecuación recursiva como se muestra a continuación.

$$\theta(k) = K(e^{-a}+1)I(k-1) + (e^{-a}+1)\theta(k-1) - e^{-a}(k-2) \tag{19}$$

donde cada instante k a través de un periodo de tiempo T se puede obtener un sistema discreto a lo largo del tiempo $t = TK$. Como en este caso se tiene como actuador un motor eléctrico se ocupará una ecuación de torque de un motor en función de la corriente eléctrico de corriente directa el cual está dado por la Ecuación (20) [61, 62].

$$\tau = nk_m i \tag{20}$$

Donde n es la relación de engranajes del motor, k_m es la constante de par del motor y la variable i es la función de corriente de excitación del motor.

4.4. Métricas de Evaluación del Modelo

Para evaluar el modelo, es necesario aplicar métricas para series de tiempo y de esta manera, poder medir el desempeño del modelo y encontrar patrones temporales en los que se pueda validar los resultados obtenidos [63].

4.4.1. Error

Indica la desviación entre el valor verdadero o deseado (y) y el obtenido ($\hat{y}$), de esta manera se puede tener magnitud y sentido del error en un instante de tiempo tiene una imagen en el intervalo $[-\infty, \infty]$, se define como:

$$e(t) = y(t) - \hat{y}(t) \tag{21}$$

4.4.2. RMSE

Es la raíz media cuadrática del error, su principal objetivo es el de medir la desviación estándar de los valores residuales entre los errores.

$$RMSE = \sqrt{\frac{1}{n_d}\sum_{i=1}^{n_d}(y_i(t) - \hat{y}_i(t))^2} \tag{22}$$

4.4.3. MAPE

El error porcentual medio absoluto, mide la distancia en porcentaje de la desviación con respecto a las mediciones, es decir, Es el promedio del error absoluto o diferencia entre el valor real y el pronóstico, expresado como un porcentaje de los valores reales.

$$MAPE = \frac{\sum_{i=1}^{n_d} (\frac{100|y_i - \hat{y}_i|}{y_i})}{n_d} \tag{23}$$

4.4.4. Coeficiente de determinación

El coeficiente de determinación o R^2, describe la proporción de varianza de la variable dependiente explicada por el modelo de predicción. Si el modelo de predicción es "perfecto", SSE es cero, y R^2 es uno. Si el modelo de predicción es un desastre, SSE es igual a SST , y no se puede explicar ninguna varianza por regresión, además R^2 es cero.

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} \tag{24}$$

$$\begin{aligned} SST &= \sum_{i=1}^{n_d} (y_i - \bar{y})^2 \\ SSR &= \sum_{i=1}^{n_d} (\hat{y}_i - \bar{\hat{y}}_i)^2 \\ SSE &= \sum_{i=1}^{n_d} (y_i - \hat{y}_i)^2 \end{aligned} \tag{25}$$

Donde $\bar{y}$ es la media aritmética de los valores reales y $\bar{\hat{y}}$ es la media aritmética de los valores de predicción

4.4.5. Autocorrelación

Indica la cantidad de relación mutua entre los valores de una serie de tiempo en un periodo determinado, además de describir alguna tendencia en la sucesión de los valores (cambios) [64, 65]. Se puede expresar como:

$$r_k = \frac{\sum_{t=1}^{n-k}(y_t - \bar{y})(y_{t+k} - \bar{y})}{\sum_{t=1}^{n}(y_t - \bar{y})^2} \tag{26}$$

Donde: y_t es la observación en el periodo t, r_k es el coeficiente de autocorrelación, $\bar{y}$ es la media de los valores de la serie, n es el numero total de periodos y y_{t+k} es la observación con k restrasos.

4.4.6. Prueba de Cointegración

Esta prueba perimétrica permite conocer la cointegración de dos series de tiempo, mediante el uso de la colinealidad de las muestras, constituyendo una combinación lineal de cada serie de tiempo [66, 67]. Esto implica que las estimaciones son libres de espurios, por lo que existe evidencia de que existe relación de las series de tiempo tanto a corto como a largo plazo.

Se plantea la siguiente hipótesis nula:

H_0: Los valores reales y las predicciones no están cointegrados.
H_1: Los valores reales y las predicciones están cointegrados.

Para realizar esta prueba es necesario realizar dos pasos, estimar los errores de las series y determinar si la serie de errores estimados son cercanos a un valor definido [68, 69].

Para poder estimar los errores en las series, se debe realizar una regresión estática de mínimos cuadrados ordinarios:

$$y_t = \hat{\alpha} + \hat{\delta} x_t + \hat{u}_t \tag{27}$$

donde $\hat{\alpha}$ y $\hat{\delta}$ son los estimadores de la regresión y $\hat{u}_t$ es el residual en el momento t. Esto permite conocer su relación potencial a largo plazo.

Posteriormente se realiza la prueba de hipótesis basada en la prueba de Dickey-Fuller (DF) [70]

$$\Delta \hat{u}_t = \lambda \hat{u}_{t-1} + \hat{\eta}_t \tag{28}$$

en donde λ es el valor de significancia de la hipótesis nula y $\hat{\eta}_t$ es un estimador de la prueba DF.

4.4.7. Prueba T-Student

Es una prueba perimétrica que permite analizar las medias de dos grupos mediante una prueba de hipotesis [71, 72]. Observando si las diferencias entre valores deseados y obtenidos (residuales) son estadísticamente significativas. Se plantea la siguiente hipotesis nula:

H_0: Los valores reales y las predicciones no son significativamente diferentes.

H_1: Los valores reales y las predicciones son significativamente diferentes.

Para realizar esta prueba se debe calcular la desviación estándar ponderada.

$$\sigma_p = \frac{\sqrt{\sum x_1^2 + \sum x_2^2}}{N_1 + N_2 - 2} \tag{29}$$

Donde en el numerador se realiza la suma de cuadrados de las muestras y $N_{1,2}$ es el tamaño de la muestra 1 y 2 respectivamente.

Una vez que se obtuvo la desviación estándar ponderada, se calcula el valor t de Student. el cual puede ser utilizado para 3 situaciones

$$\nabla_1 x_{1,2} = \bar{x} - \mu \quad \nabla_2 x_{1,2} = \bar{x}_1 - \bar{x}_2 \quad \nabla_3 x_{1,2} = \bar{x}_{1j} - \bar{x}_{2j} \tag{30}$$

Donde $\nabla_1 x_{1,2}$ representa una sola muestra que se compara con la población del mismo grupo, $\nabla_2 x_{1,2}$ representa la comparación de dos muestras independientes del grupo y $\nabla_3 x_{1,2}$ representa dos muestras relacionadas, es decir, siendo la misma muestra pero medida en dos momentos diferentes.

$$t = \frac{\nabla_b}{\frac{\sigma_p}{\sqrt{N}}} \tag{31}$$

Indicando el valor t la cantidad de unidades estándares que están separando las medias de los dos grupos.

4.4.8. Deformación Dinámica de Tiempo (DTW)

Es una técnica para poder calcular la similitud entre 2 series de tiempo de diferente longitud [73].

Se define a partir de dos series de tiempo $X : \{x_1, x_2, \cdots, x_n\}$ y $Y : \{y_1, y_2, \cdots, y_m\}$. Donde el objetivo es obtener un coeficiente de similitud DTW y un camino

deformado (*warping path* w_p, véase Figura 13) [74].

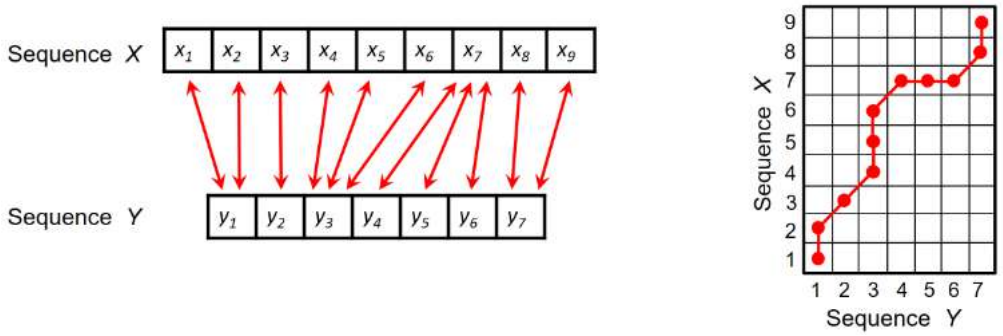


Figura 13: Warping path de la secuencia del algoritmo DTW [74].

Para obtener la similitud y el camino, hay que considerar una matriz de costo $C \in \mathbb{R}^{n \times m}$ de la función DTW en donde se debe obtener el costo total c_p.

$$c_p := \sum_{\ell=1}^{L} c(x_{n\ell}, y_{m\ell}) = \sum_{\ell=1}^{L} C(n_\ell, m_\ell) \tag{32}$$

Para posteriormente calcular el valor DTW de cada combinación de elementos $DTW(n,m)$.

$$DTW(X,Y) := c_p^*(X,Y) = \{\min c_p(X,Y) | P \text{ es un } (n,m) - w_p\} \tag{33}$$

4.5. Algoritmo de Límite Superior de Confianza (UCB)

El algoritmo UCB (*Upper Condifence Bound*) permite explorar, tomando en cuenta los riesgos de tomar nuevas acciones y sus recompensas, las cuales, pueden ser desfavorables para el rendimiento del modelo [75]. Se suele utilizar para poder tomar siempre las mejores nuevas decisiones a partir de la distribución de la recompensa de los valores de la acción [76].

$$A_t = \text{argmax}\left(Qt(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}}\right) \tag{34}$$

Donde A_t es la acción a tomar, c es una constante de exploración ($0 < c \leq 2$), $Q_t(a)$ es el valor Q para la acción a en el *timestep* t y $N_t(a)$ es la cantidad de veces que se ha visitado dicho estado-acción. Teniendo como principal ventaja, la de poder aprender en ambientes multivariados y estocásticos.

5. Hipótesis

La implementación de algoritmos de control inteligente basados en métodos de aprendizaje por refuerzo permitirán estimar las acciones para controlar la posición de una para robótica de 3 grados de libertad.

6. Objetivos

6.1. Objetivo General

Desarrollar y evaluar un algoritmo de aprendizaje automático para controlar la posición de los eslabones del modelo de pata robótica de un hexápodo con 3 grados de libertad.

6.2. Objetivos Específicos

- Obtener modelo dinámico del comportamiento de la pata a través de investigación documental y análisis dinámico de sistemas.

- Desarrollar un ambiente de simulación para realizar la experimentación requerida mediante software.
- Crear algoritmos de aprendizaje por refuerzo con variaciones en sus elementos utilizando herramientas de software para observar su comportamiento en el ambiente y estimar hiperparámetros.
- Realizar análisis de pruebas de los algoritmos mediante seguimiento de trayectorias, métricas de desempeño y pruebas de hipótesis para comprobar los resultados.

7. Metodología

Se propone una metodología basada en la creación del modelo y ambiente, la exploración de políticas y el análisis de desempeño del modelo interactuando con el ambiente (véase Figura 14). En donde se inicia con una exploración del estado del arte y revisión documentada de aprendizaje por refuerzo y los sistemas no lineales que involucran al ambiente. Planteando la creación de un ambiente basado en un modelo Markoviano (véase Figura 5), así como también, su base de datos. Posteriormente se realiza la construcción de un modelo de aprendizaje por refuerzo (RL), en donde se seleccionará una política y sus hiperparámetros. Evaluando y verificando el desempeño del modelo con dicha política. Finalmente se realiza el análisis de resultados para comprobar o refutar las hipótesis planteadas.

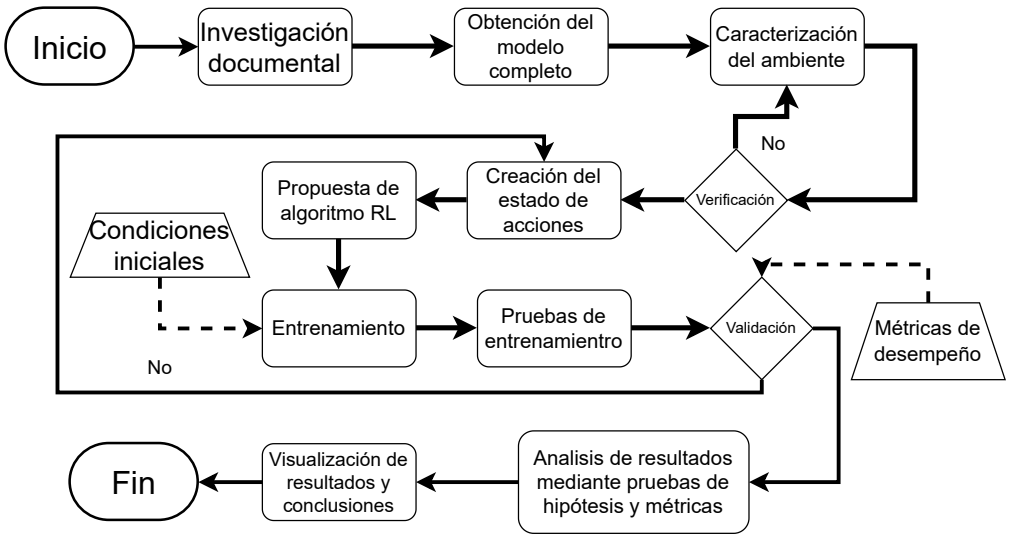


Figura 14: Metodología propuesta para la construcción y evaluación del ambiente y el modelo de aprendizaje por refuerzo (creación propia).

7.1. Sistema de Control

A continuación se muestra el sistema de control propuesto, en donde se destaca la codificación y el tratamiento de los datos a través de cada bloque para que pueda funcionar de manera correcta (igualando los niveles granulares de cada mundo). Además de que se puede observar el comportamiento del flujo de los datos (véase Figura 15).

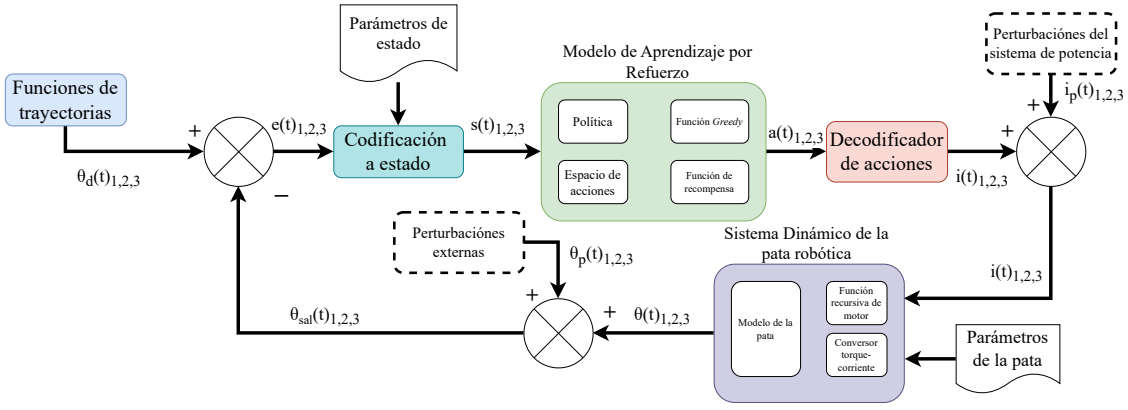


Figura 15: Sistema de control del sistema completo (creación propia).

En donde se inicia por el calculo del error $e(t)_{1,2,3}$ para el instante (t) en los 3 grados de libertad, siendo un atributo continuo, se realiza una codificación (cambio de nivel de granularidad a un nivel menos abstracto) para obtener un valor de estado para cada grado de libertad $s(t)_{1,2,3}$ (atributo categórico ordinal), mediante el uso de los parámetros de estado es que se hace el criterio de codificación. Con ese nivel de estado es posible que el modelo de aprendizaje por refuerzo pueda realizar su procesamiento de datos y toma de decisiones para cada grado de libertad $a(t)_{1,2,3}$ (atributo categórico ordinal), cabe resaltar que un único modelo de aprendizaje por refuerzo controla los 3 grados de libertad. Posteriormente las acciones son decodificadas a un valor de corriente $i(t)_{1,2,3}$ (atributo continuo), los cuales pueden ser afectadas por perturbaciones aleatorias provenientes del sistema de potencia. Esta corriente es aplicada al sistema dinámico de la pata robótica y permite generar un movimiento que se convierte en un ángulo de salida para cada grado de libertad, este puede ser afectado por condiciones externas desconocidas y aleatorias (fricción anormal en los eslabones, obstáculos, etc.). Finalmente este ángulo de salida es utilizado para calcular el error y pasar al siguiente instante t.

7.2. Selección de políticas

La metodología para la selección y prueba de políticas se basa en el diseño y observación del ambiente y del modelo de aprendizaje por refuerzo basado en MDP. En donde se analizan las variables del ambiente y sus métricas de evaluación mencionadas con anterioridad. En la Figura 16, se observa el ciclo de selección y evaluación de los modelos de aprendizaje por refuerzo.

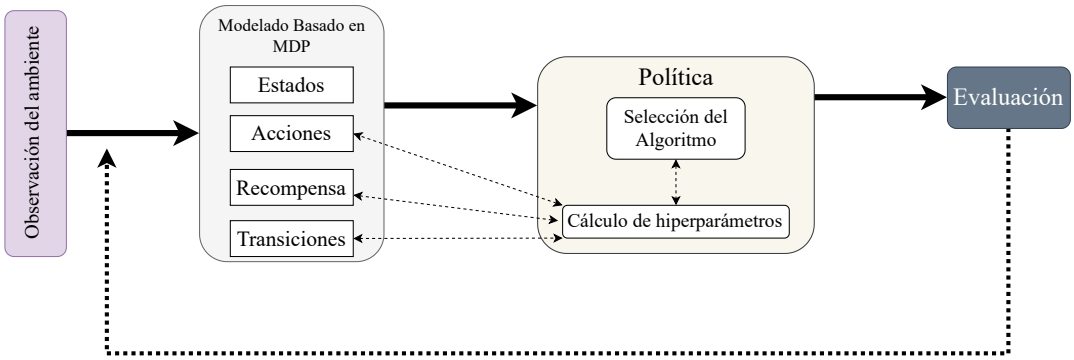


Figura 16: Subproceso de selección de algoritmos de aprendizaje por refuerzo [54].

De una manera más especifica se observa en la Figura 17, el modo en que se incrustan los algoritmos basados en tablas Q (*Q-Learning*), puesto que tienen el mismo funcionamiento, variando en su función de aprendizaje.

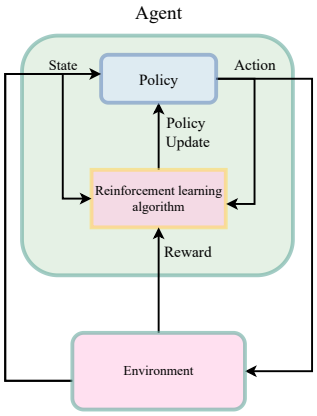


Figura 17: Modelo de aprendizaje por refuerzo con un agente basado en aprendizaje Q [77].

El modelo de aprendizaje Q, se basa en un comportamiento como el que se muestra en la Figura 18.

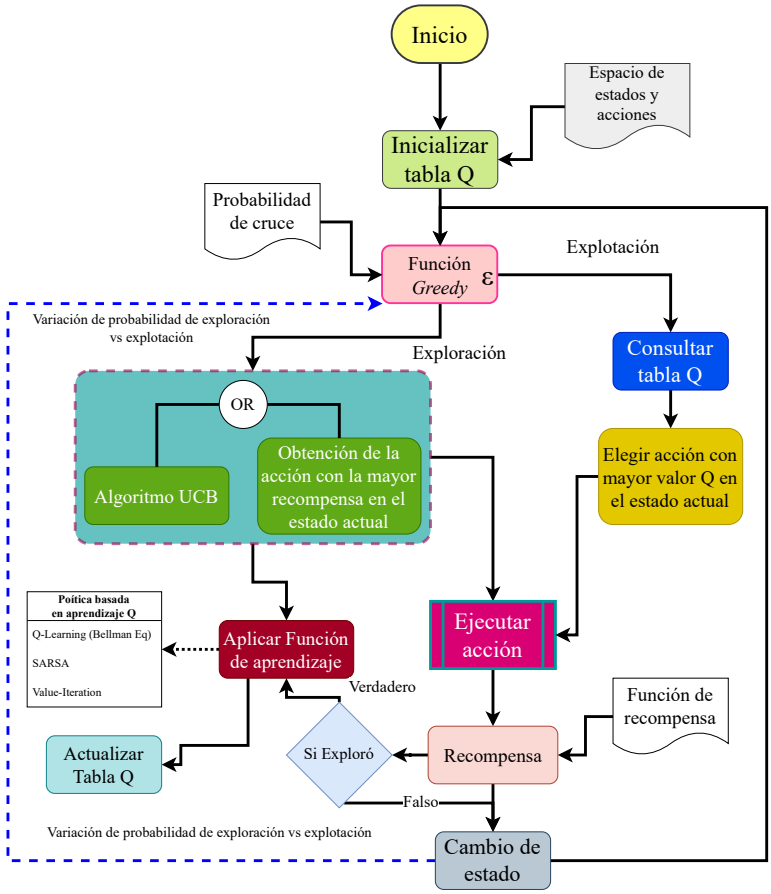


Figura 18: Diagrama de flujo de comportamiento de algoritmo basado en aprendizaje Q (creación propia).

Este modelo del comportamiento basado en aprendizaje Q, en donde se inicia con una tabla Q con valores 0 o aleatorios muy pequeños. Posteriormente, consta de una función ε que controla el porcentaje de ocasiones en que se explora y se explota, con ayuda de un valor aleatorio que cambia cada vez que ocurre un paso, estos dos valores (ε y probabilidad de cruce P_c) se comparan y según un criterio se decide explorar o explotar. En caso de explorar se pueden realizar 2 diferentes métodos de exploración. El primero realiza un barrido de todas las posibles acciones y se recolectan sus recompensas, eligiendo la acción que tenga la mayor recompensa. Por otro lado se tiene el algoritmo UCB que se basa en la Ecuación (34). Para finalmente modificar el valor Q en

la posición del estado-acción correspondiente con la acción elegida.

Por otro lado, si el criterio elige explotar, se realiza una consulta en la tabla Q y se elegirá la acción con el mayor valor Q en el estado en el que se encuentra. La acción tomada por la decisión de explorar o explotar modificará el estado actual del ambiente, generando un cambio de estado y repitiendo el ciclo en la función ε, la cual, podrá cambiar su valor según el criterio que se requiera.

7.3. Construcción del Modelo de Aprendizaje por Refuerzo

Una vez que se conocieron las principales estructuras del modelo general de aprendizaje por refuerzo, se observó que hay muchas combinaciones con las que es posible acoplar el modelo. Por lo que se creó una métodología para construir el modelo en la Figura 19 donde se puede apreciar el proceso de elección de cada elemento que es necesario.

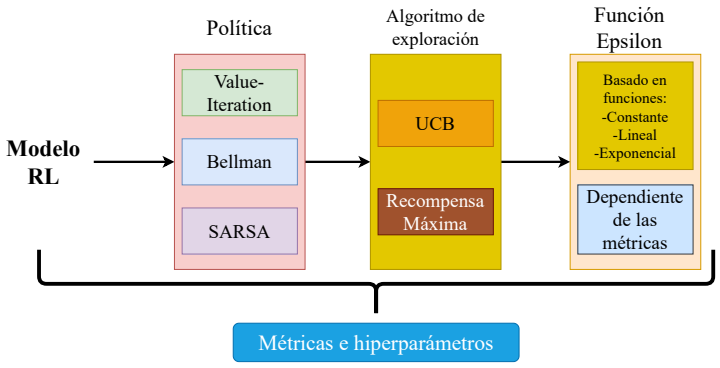


Figura 19: Proceso de construcción del modelo de aprendizaje por refuerzo (creación propia).

7.4. Metodología de entrenamiento y base de datos

Una vez conocido el modelo de aprendizaje del algoritmo basado en tablas Q, se propone una metodología de aprendizaje con base a las trayectorias

deseadas. Se generaron datos sintéticos basado en las trayectorias originales (véase Figura 20) debido a la falta de datos, con el fin de que pueda explorar la mayor cantidad de posibilidades y variaciones a las trayectorias originales, obteniendo más patrones de entrenamiento y no solamente 3. Para la generación de los datos sintéticos, se propusieron las funciones originales con ruido aleatorio distribuido uniformemente.

$$\theta_{1,2,3}(t) = f_{1,2,3}(t) + rand \tag{35}$$

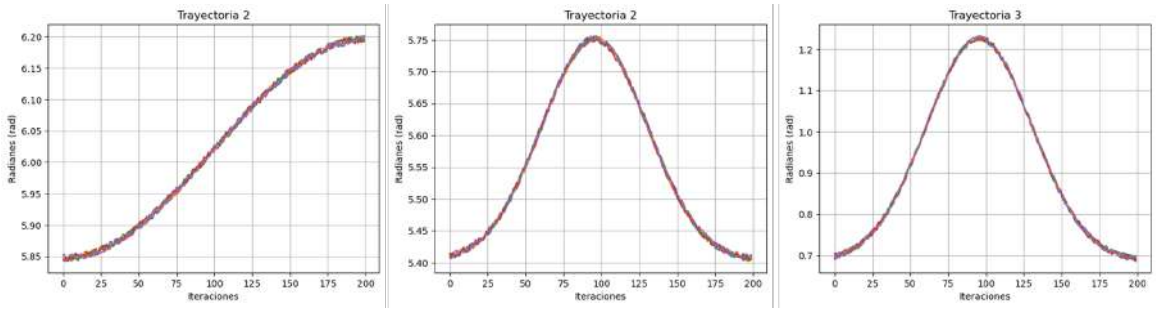


Figura 20: Datos sintéticos de las trayectorias originales (creación propia).

La propuesta de entrenamiento consta de un proceso de aprendizaje basado en un 100 % de exploración, utilizando los 3 componentes de los grados de libertad en sus datos sintéticos y finalmente la prueba con las trayectorias original ejecutadas una cierta cantidad de ocasiones, con el fin de observar su estabilidad ante tareas repetitivas y el comportamiento de su función ε a lo largo de cada episodio.

8. Resultados y Discusión

8.1. Funciones de Trayectorias Generalizadas

Debido a la complejidad del calculo de las trayectorias $\theta_{2,3}$ representado en las Ecuaciones (13) (14), en su función $k1$ que es muy complicada de proponer correctamente debido a factores más específicos del modelo matemático (mostradas en los Anexos). Se realizó una caracterización de dichas trayectorias haciendo la siguiente aproximación Gaussiana en un dominio $0 \leq \zeta \leq 2\pi$ (véase Figura 21).

$$\begin{aligned} \theta_2(\zeta) &\approx f_2(\zeta) + d\beta + f_2(0), \quad f_2(\zeta) = a_1 e^{\frac{(\zeta - b_1)^2}{2c_1^2}} \\ \theta_3(\zeta) &\approx f_3(\zeta) + d\chi + f_3(0), \quad f_3(\zeta) = a_2 e^{\frac{(\zeta - b_2)^2}{2c_2^2}} \end{aligned} \tag{36}$$

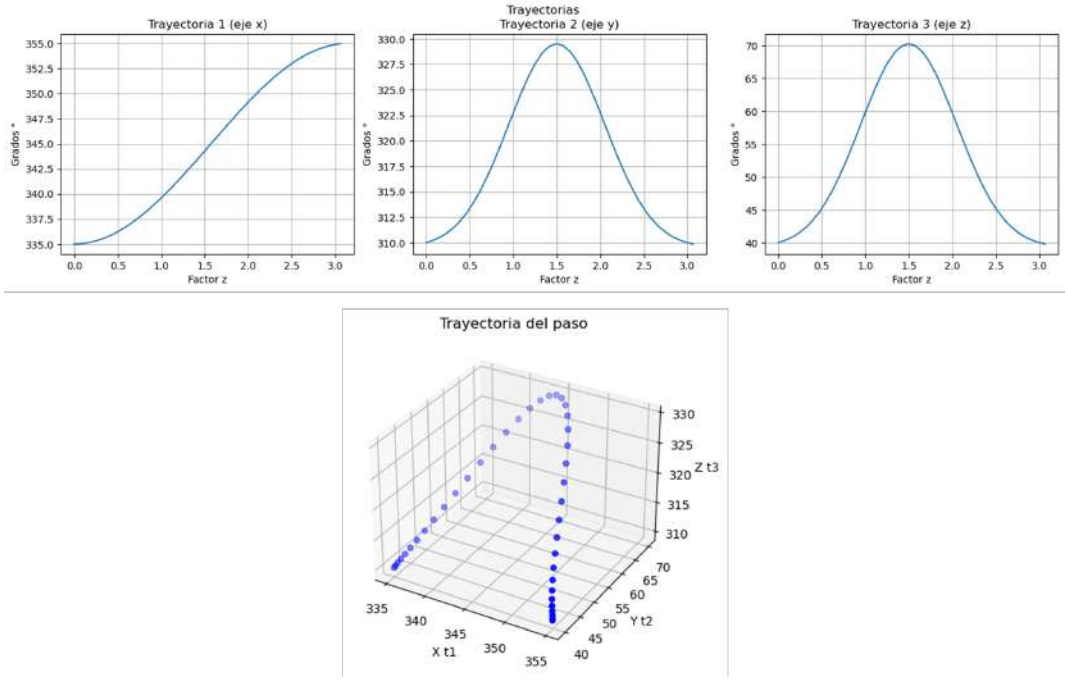


Figura 21: Trayectorias Gaussianas generalizadas (creación propia).

Obteniendo una función más homogénea y simétrica en la trayectoria del paso. De esta forma es posible trabajar con funciones continuas de trayectorias, teniendo como parámetros iniciales de la pata robótica (véase nomenclatura completa en Anexos: Funciones de la Matriz de Torque).

$$\begin{array}{ccccc} m_1 = 0.3, & J_1 = 1, & L_1 = 0.2, & A\phi = 335, & B\phi = 10 \\ m_2 = 0.5, & J_2 = 10, & L_2 = 0.3, & A\beta = 4, & B\beta = 40 \\ m_3 = 2, & J_3 = 1, & L_3 = 0.1, & A\chi = 310, & B\chi = 310 \\ a_1 = 31, & b_1 = 1.5, & c_1 = 0.55, & a_2 = 20, & b_2 = 1.5 \\ c_2 = 0.55, & K_{1,2,3} = 7.35, & a_{1,2,3} = 0.14 & & \end{array} \tag{37}$$

8.2. Construcción del Espacio de Estados y Acciones

Se propone un cambio granular de las variables con las que se está trabajando principalmente (corriente de salida i,posición θ), con el fin de poder crear un espacio de estados y acciones normalizado. Para realizar el espacio de acciones se proponen las siguientes reglas.

Para toda corriente i en el intervalo $[a_i, b_i]$, en donde se tiene una pertenencia a una acción normalizada A secuencialmente hasta n acciones.

$$A : \{A_1, A_2, \cdots, A_n\} \quad i \in [a_i, b_i] \tag{38}$$

Entonces para calcular la una acción derivada de una corriente i_t a una acción A_t se debe calcular encontrando la posición en la que se encuentra dentro del intervalo de cada acción:

$$
\begin{aligned}
t &< n \\
t &= \arg([A_1, A_n]) \\
A_t &= \left[t\tfrac{b_i - a_1}{n}, (t+1)\tfrac{b_i - a_i}{n} \right]
\end{aligned}
\tag{39}
$$

De igual manera es posible expresar el espacio de estados mediante este concepto (véase Figura 22). Sin embargo, se realizó un ambiente basado en el error del sistema calculado restando el valor deseado con el obtenido en el paso t $e(t) = \theta(t)_d - \theta(t)_{sal}$, con un intervalo $[a_e, b_e]$.

$$
\begin{aligned}
S &: \{S_1, S_2, \cdots, S_m\} \\
S_t &= \arg(e(t) \in [a_e, b_e] | S)
\end{aligned}
\tag{40}
$$

Figura 22: Diagrama de espacio de estados (creación propia).

8.2.1. Verificación del Ambiente

La construcción del espacio de estados y acciones debe ser verificada mediante un análisis de las formulas planteadas para la construcción del espacio de estados y acciones (Ecuaciones (39)(40)). Por lo que gráficamente se debe realizar una discretización de los estados y acciones, para convertir las funciones de trayectorias y de corriente continuas en datos discretos basados en estados obtenidos a partir de intervalos (véase Figura 23). Obteniendo una cantidad de iteraciones que representan a la función continua, representada como t_s o *timesteps*.

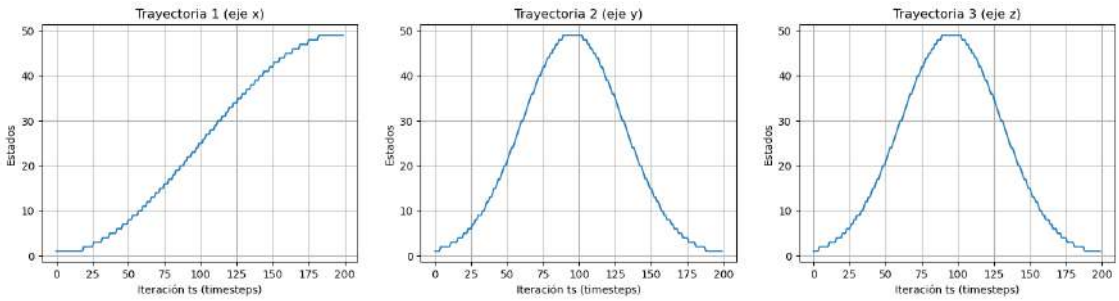


Figura 23: Verificación del ambiente (creación propia).

8.3. Elección de Hiperparámetros

8.3.1. Factor de aprendizaje y Tasa de Descuento en un Ambiente Basado en cada Grado de Libertad

Se realizó la prueba de la rejilla, en donde los hiperparámetros se variarán de manera ligera, con el fin de obtener los hiperparámetros α, γ donde obtengan el menor valor RMSE (es decir, donde haya mayor concordancia de la trayectoria con la salida). Dicha prueba consta de 35000 episodios en una tabla Q de dimensión (20,50,3) (20 estados, 50 acciones y 3 grados de libertad), aplicando una función ε linear descendente (véase Figura 10). Obteniendo las siguientes regiones demarcadas en la Figura 24.

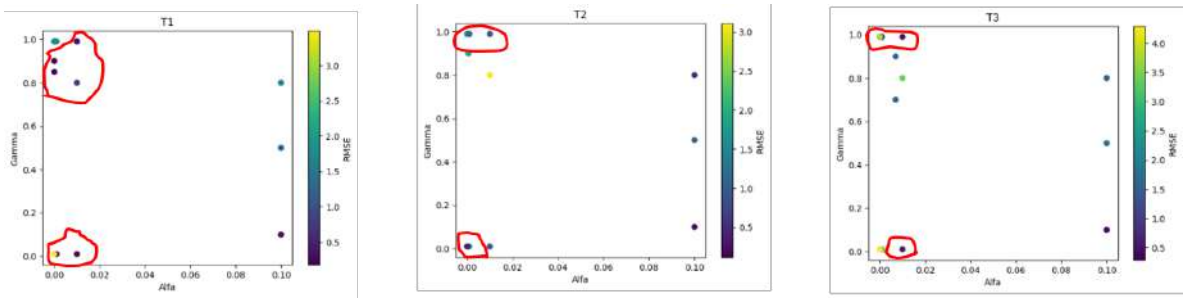


Figura 24: Método de la rejilla para cada grado de libertad (creación propia).

En donde se observa una tendencia a que se encuentran los mejores valores α, γ en la misma región en los 3 grados de libertad.

$$\begin{aligned} \gamma_{1,2,3} &\to 1^- \\ \alpha_{1,2,3} &\to 0^+ \end{aligned} \tag{41}$$

Por lo que los valores del factor de aprendizaje α tienen que ser muy pequeños y positivos, lo que significa que tendrá pocas variaciones en su aprendizaje y esto le permitirá aprender poco a poco sin dar saltos grandes, por otro lado, la tasa de descuento γ alta ayudará a darle más importancia a la acción actual con base en las futuras.

Sin embargo, hay que tomar en cuenta la aleatoriedad del espacio de acción, lo que conlleva a que se puedan tomar diferentes acciones en los mismos estados, incentivando el aprendizaje del modelo, pero evitando una similitud entre valores muy cercanos, esto es posible de atenuar usando métodos de ensamble que permitirán, gracias al teorema de los grandes números, estabilizar y generar tendencias en las decisiones a tomar. Cabe mencionar que los tiempos de ejecución son bastante tardados, por lo que este método no es muy recomendable cuando se tienen espacios demasiado amplios, en este caso fue posible identificar las zonas de menor RMSE mediante los principios de aprendizaje automático (Machine Learning).

Una vez que se obtienen los valores con el método de la rejilla, se pueden ir juntando funciones *Greedy* ε con los resultados de la tabla Q para mejorar el desempeño del modelo. Por lo que se realizó un conjunto de pruebas subsecuentes con diferentes funciones ε en un método de ensamble y *transfer learning* (véase Figura 25).

Figura 25: Método de ensamble de modelos (creación propia).

Este proceso de ensamble se aplicará a los mismos 35000 episodios y será de utilidad para el aprendizaje del algoritmo y actualizar la tabla Q, teniendo valores de referencia ya establecidos del proceso anterior y finalmente observar el comportamiento de las principales métricas conforme se aplica cada proceso de ensamble (véase Figura 26).

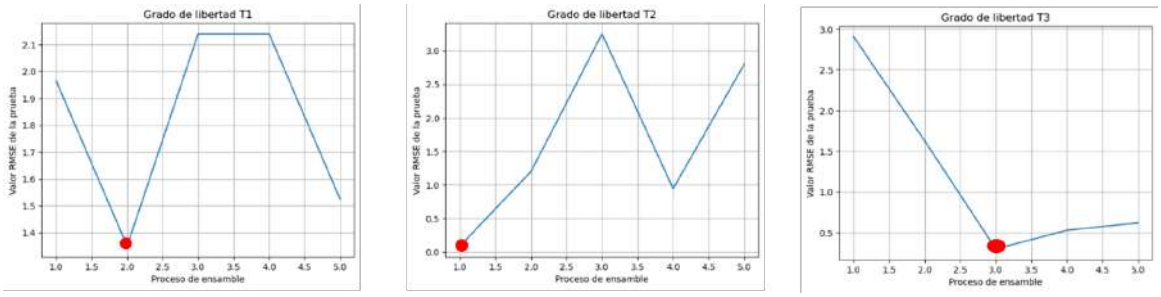


Figura 26: Desempeño de método de ensamble para cada grado de libertad (creación propia).

Es posible observar que algunas salidas de cada grado de libertad tienden a empeorar conforme se avanza el proceso de ensamble, eso puede deberse a un sobre aprendizaje o a que empieza a aprender mal algunas cosas debido a la cantidad de información que se proporciona. Por lo que se tomó la tabla Q de la prueba de desempeño de cada grado de libertad en donde se tuviera el mejor RMSE y de esta manera tener la mejor tabla Q para cada ambiente en el espacio de estados correspondiente.

Una vez obtenidos los hiperparámetros que mejor coincidían con los mejores valores RMSE y aplicando métodos de ensamble para observar si hay mejoría conforme continúa recibiendo datos, se realiza una prueba con un

Greedy $\varepsilon = 0.95$, lo que significa que hay un 95 % de probabilidad de explotación y un 5 % de exploración. Se extrajeron las tablas Q con el mejor RMSE, obteniendo los siguientes resultados de desempeño en la Tabla 2.

Tabla 2: Desempeño evaluado con métricas para cada grado de libertad.

Grado de libertad	RMSE (rad)	MAPE(%)
θ_1	0.638	8.076
θ_2	0.09	1.3138
θ_3	0.304	26.66

Como se observa el segundo grado de libertad es el que mejor se desempeña y el tercer grado de libertad es el que peor se desempeña en la métrica MAPE, debido a que se toma en cuenta el error porcentual debido al rango de cada trayectoria $\Delta\theta_1 = 0.34$ rad, $\Delta\theta_2 = 0.3375$ rad, $\Delta\theta_3 = 0.52$ rad. Por lo que, al sobrepasar ese valor, incrementa en mayor medida el error porcentual. Sin embargo, numéricamente no se desvían tanto.

En la Figura 27 se observa que las trayectorias definidas por el modelo (naranja) visualmente no tienen mucha coincidencia con la trayectoria deseada (azul). Por lo que, aunque las métricas calculadas en la Tabla 2 no son tan malas, para la locomoción de la pata robótica son muy inexactas.

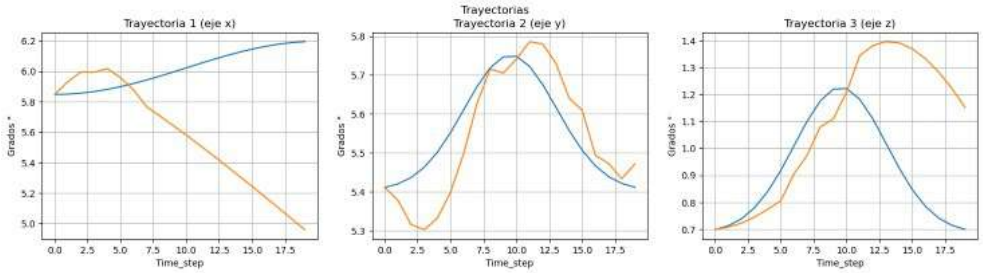


Figura 27: Desempeño de método de ensamble con el menor RMSE (creación propia).

Esto implica que tendrá un movimiento muy errado y poco coordinado en conjunto (véase Figura 28).

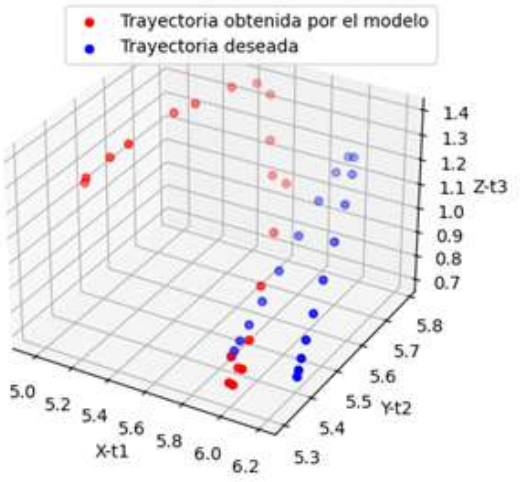


Figura 28: Movimiento tridimensional de la pata robótica con método de ensamble (creación propia).

El algoritmo de aprendizaje por refuerzo basado en aprendizaje Q se puede utilizar para la toma de decisiones de las trayectorias de la pata robótica. Sin embargo, la elección de los hiperparámetros del modelo y la construcción de un ambiente que caracterice adecuadamente el problema es decisivo para el desempeño adecuado del sistema de control, lo cual se ve aún más afectado por la aleatoriedad de la función ε.

8.3.2. Estados, Acciones e Iteraciones

Se realizaron pruebas modificando los 3 hiperparámetros con mayor incertidumbre (número de estados S, número de acciones A y numero de pasos o iteraciones t_s en diferentes métodos de exploración). Los hiperparámetros γ y α fueron aproximados mediante el método de la rejilla (Véase Figura 24) y expresados en la Ecuación (41). Todo esto con el fin de encontrar la dimensión adecuada para la tabla de aprendizaje Q (véase Tabla 3).

Tabla 3: Análisis exploratorio de diferentes tablas de aprendizaje (creación propia).

Política	Método de exploración	Hiperparámetros	Estados-Acciones	Tiempo de Entrenamiento (min)	Porcentaje de exploración de la tabla Q
Value-Iteration	Recompensa Máxima $\max(r(s_t, a_t))$	$\gamma = 0.99$ $\alpha = 0.01$ $E_t = 500$ $t_s = 200$ $\epsilon_t = 0$	$S = 1002 \quad A = 1000$	5	6.13
			$S = 502 \quad A = 500$	3	21
			$S = 202 \quad A = 200$	1.5	64.12
			$S = 102 \quad A = 100$	0.63	82.39
			$S = 52 \quad A = 200$	1.16	81.43
	Algoritmo UCB	$\gamma = 0.99$ $\alpha = 0.005$ $E_t = 1000$ $t_s = 200$ $\epsilon_t = 0$	$S = 1002 \quad A = 1000$	0.1825	3.63
			$S = 502 \quad A = 500$	0.095	0.34
			$S = 202 \quad A = 200$	0.088	1
			$S = 102 \quad A = 100$	0.085	4.8
			$S = 52 \quad A = 200$	0.0845	21.4

Se puede observar que el tiempo de entrenamiento es bajo, debido a la cantidad baja de épocas de entrenamiento E_t, las cuales permitieron observar un comportamiento del índice de la exploración de las combinaciones estado-acción. Mostrando que el algoritmo UCB es mucho más veloz de ejecutar. Sin embargo, es poco exploratorio (véase Figura 29).

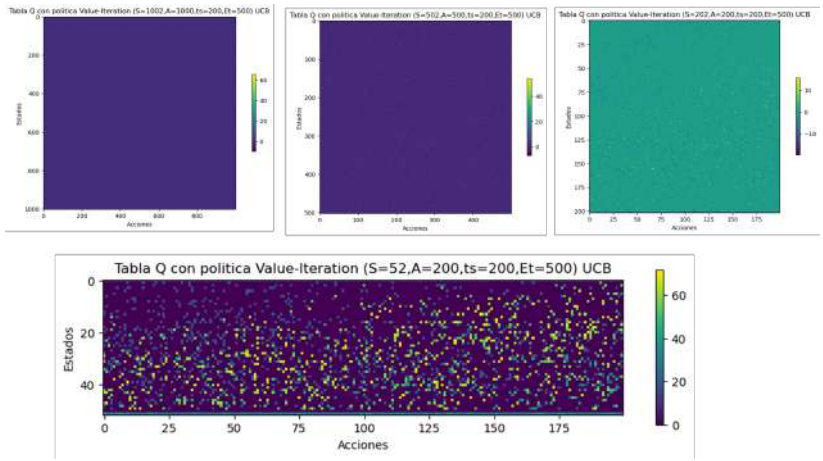


Figura 29: Espacio estado-acción con el método de exploración UCB(creación propia).

Por otro lado, se tiene el algoritmo de exploración basado en la recompensa máxima, el cual permite realizar una mayor cantidad de exploración en la tabla de estado-acción, logrando un gráfico de tendencia mucho más marcado conforme se va decrementando la cantidad de estados y acciones. Indicando

un aprendizaje más fuerte como se muestra en la Figura 30.

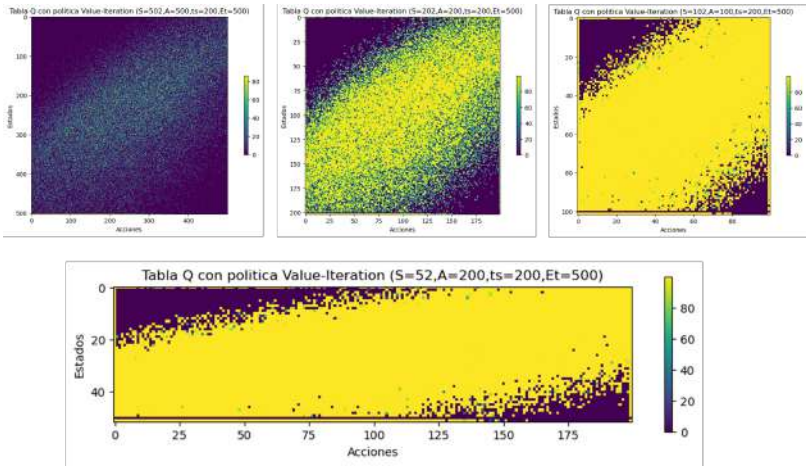


Figura 30: Espacio de aprendizaje estado-acción con el método de exploración basado en máxima recompensa (creación propia).

Por lo que gracias a esta prueba, se definió que la cantidad de estados y acciones debe encontrarse en el intervalo $20 > S < 100$ y $75 > A < 250$.

8.4. Función ε Propuesta

Para poder realizar un equilibrio adecuado entre explorar y explotar, se propone una función ε_p, la cual depende de la recompensa promedio del episodio $\bar{r}(E)$ y la tasa de retorno $G_t(E)$ divididas por un peso propuesto ω_1, ω_2. De este modo, cuando se encuentre una disminución en el rendimiento de estas métricas, se realizará un incremento en la exploración, tomando en cuenta nuevas opciones que podrían mejorar el rendimiento del modelo.

$$\varepsilon_p(E) = \frac{1}{1 + \omega_1 e^{-(\omega_2 \bar{r}(E)^3 - \omega_3 G_t(E)^2 + \omega_4)}} \tag{42}$$

Gráficamente es una función sigmoide (función logística generalizada) tridimensional con diferente suavidad dependiendo del eje en el que se encuentren

(véase Figura 31). Esta suavidad se define por los coeficientes $\omega_1, \omega_2, \omega_3, \omega_4$. En donde se destaca que su dominio en las variables independientes es $\bar{r}, G_t$: $(-\infty, \infty)$ y el rango de la variable $\varepsilon : [0, 1]$ donde 0 es 100 % explorar y 1 es 100 % explotar.

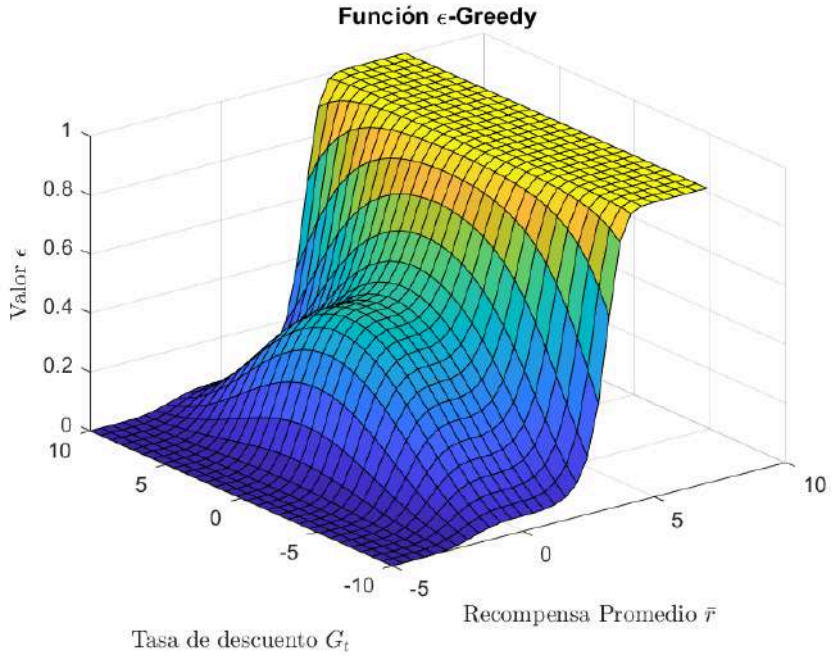


Figura 31: Función exploración/explotación propuesta (creación propia).

Obteniendo una buena caracterización del espacio de las métricas mencionadas anteriormente debido a la naturaleza del algoritmo de aprendizaje por refuerzo. Siguiendo un comportamiento de exploración (aprendizaje) cuando empeoren sus métricas y un comportamiento de explotación (acciones propias) cuando sus métricas sean lo suficientemente buenas para que pueda desempeñarse mediante su tabla de aprendizaje Q. Todo esto permite realizar un equilibrio entre explorar y explotar, evitando sobreaprendizaje o subaprendizaje.

8.5. Entrenamiento y Pruebas del Modelo en el Ambiente Basado en Error

Se realizaron diferentes pruebas de entrenamiento según la política y el método de exploración en el sistema compensado (lazo cerrado) con un número de episodios de entrenamiento $E_t = 1000$ con ruido basadas en la Ecuación (35). Los principales comportamientos del modelo fueron descritos en la Tabla 4. En donde se destaca que en todos los modelos tienen $S = 52, A = 200, \varepsilon = 0, t_s = 200$. Se destaca que los tiempos de entrenamiento son muy altos en las pruebas con un método de exploración basado en recompensa máxima. Por otro lado, se observó que en los métodos basados en UCB tuvieron un porcentaje mayor de exploración. Esto debido a la baja cantidad de E_t. Sin embargo, cabe destacar que a largo plazo los algoritmos basados en recompensa máxima, tienen una exploración mayor dependiendo del sistema. Finalmente, las métricas de aprendizaje en el método de recompensa máxima son menores, puesto que siempre explora todas las acciones del estado y toma la mejor.

Tabla 4: Resultados del entrenamiento (creación propia).

Prueba	Política	Método de Exploración	Hiperparámetros	Función de Recompensa	Métricas de aprendizaje	Tiempo de entrenamiento	Porcentaje de exploración
1	Value-Iteration	Recompensa máxima	$\alpha = 0.01$	$r(t)\begin{cases}-(100e(t)^2)+1 \; si \; e(t) \geq -1 \\ -10 \; si \; \lvert e(t)\rvert \geq 0.1\end{cases}$	$\bar{r} = -0.39 \pm 1.71 \quad \bar{G}_t = -120.83 \pm 244.69$	123.46s	12.7
2	Bellman		$\alpha = 0.009$ $\gamma = 0.99$		$\bar{r} = -0.39 \pm 1.71 \quad \bar{G}_t = -120.83 \pm 244.80$	120.9s	12.7
3	SARSA				$\bar{r} = -0.39 \pm 1.71 \quad \bar{G}_t = -120.83 \pm 244.70$	120.17s	12.7
4	Value-Iteration	UCB $c = 1$	$\alpha = 0.01$	$r(t)\begin{cases}-(100e(t)^2)+1 \; si \; e(t) \geq -1 \\ -10 \; si \; \lvert e(t)\rvert \geq 0.1\end{cases}$	$\bar{r} = -9.36 \pm 0.78 \quad \bar{G}_t = -768.24 \pm 106.58$	4.37s	66.95
5	Bellman		$\alpha = 0.009$ $\gamma = 0.99$		$\bar{r} = -9.36 \pm 0.78 \quad \bar{G}_t = -768.24 \pm 106.58$	4.1s	66.26
6	SARSA				$\bar{r} = -9.34 \pm 0.88 \quad \bar{G}_t = -766.46 \pm 106.58$	3.66s	69.26

Posteriormente con las tablas Q entrenadas anteriormente, se generaron episodios de prueba E_p basados en las funciones originales $\theta_{1,2,3}$ (transferencia de aprendizaje). En donde se observa el desempeño con las métricas de

evaluación en la Tabla 5. En donde se utilizó $\omega_1 = 4, \omega_2 = 0.9, \omega_3 = 0.6, \omega_4 = 2$.

Los modelos que mejor desempeño general tuvieron fueron los basados en SARSA, puesto que fueron los que tuvieron los mejores valores *RMSE*, R^2 y *MAPE* en $\theta_{2,3}$. También fueron los modelos con un menor valor I_{RMS} implicando un menor consumo energético. Por otro lado los modelos basados en la ecuación de Bellman (*Q-Learning*) fueron los que tuvieron un mejor desempeño en θ_1 y el menor tiempo de computo por ejecución de la acción, siendo este, el más veloz computacionalmente hablando. Adicionalmente se debe resaltar que los episodios de prueba y entrenamiento fueron menores y más cortos en comparación a los presentados en el estado del arte.

Tabla 5: Resultados de las pruebas con transferencia de aprendizaje (creación propia).

Prueba	Episodios	Método de exploración	Función ϵ	$RMSE$ (rad)	R^2	$MAPE$ %	$\bar{I}_{RMS}$(mA)	Tiempo de ejecución de la acción (s)
1	$E_{p2} = 1800$	Recompensa Máxima	$\epsilon = \frac{1}{1+\omega_1 e^{-(\omega_2 r^3 - \omega_3 v^2 + \omega_4)}}$	$\theta_1 = 0.004$ $\theta_2 = 0.006$ $\theta_3 = 0.006$	$\theta_1 = 0.998$ $\theta_2 = 0.997$ $\theta_3 = 0.998$	$\theta_1 = 0.06$ $\theta_2 = 0.09$ $\theta_3 = 0.6$	$\theta_1 = 5 \pm 1.1$ $\theta_2 = 5 \pm 1.1$ $\theta_3 = 5 \pm 1$	$3.5 \times 10^{-4} \pm 1.9 \times 10^{-3}$
2	$E_{p2} = 900$	Recompensa Máxima		$\theta_1 = 0.0007$ $\theta_2 = 0.0077$ $\theta_3 = 0.007$	$\theta_1 = 0.999$ $\theta_2 = 0.997$ $\theta_3 = 0.998$	$\theta_1 = 0.01$ $\theta_2 = 0.01$ $\theta_3 = 0.68$	$\theta_1 = 2.1 \pm 1.6$ $\theta_2 = 4 \pm .8$ $\theta_3 = 5.3 \pm .5$	$3.5 \times 10^{-4} \pm 2 \times 10^{-3}$
3	$E_{p2} = 900$	Recompensa Máxima		$\theta_1 = 0.007$ $\theta_2 = 0.005$ $\theta_3 = 0.007$	$\theta_1 = 0.996$ $\theta_2 = 0.997$ $\theta_3 = 0.998$	$\theta_1 = 0.09$ $\theta_2 = 0.08$ $\theta_3 = 0.65$	$\theta_1 = 3.3 \pm 1$ $\theta_2 = 3.7 \pm .83$ $\theta_3 = 4.4 \pm .5$	$3.4 \times 10^{-4} \pm 2.2 \times 10^{-3}$
4	$E_{p2} = 900$	UCB $c = 1$		$\theta_1 = 0.004$ $\theta_2 = 0.005$ $\theta_3 = 0.006$	$\theta_1 = 0.998$ $\theta_2 = 0.998$ $\theta_3 = 0.998$	$\theta_1 = 0.06$ $\theta_2 = 0.07$ $\theta_3 = 0.57$	$\theta_1 = 5.7 \pm 2.2$ $\theta_2 = 5.8 \pm 2.3$ $\theta_3 = 5.8 \pm 2.3$	$5.2 \times 10^{-5} \pm 8 \times 10^{-4}$
5	$E_{p2} = 1800$	UCB $c = 1$		$\theta_1 = 0.005$ $\theta_2 = 0.005$ $\theta_3 = 0.006$	$\theta_1 = 0.99$ $\theta_2 = 0.99$ $\theta_3 = 0.998$	$\theta_1 = 0.07$ $\theta_2 = 0.07$ $\theta_3 = 0.6$	$\theta_1 = 6.9 \pm 1.3$ $\theta_2 = 6.9 \pm 1.3$ $\theta_3 = 6.8 \pm 1.3$	$4.15 \times 10^{-5} \pm 7 \times 10^{-4}$
6	$E_{p2} = 900$	UCB $c = 1$		$\theta_1 = 0.003$ $\theta_2 = 0.004$ $\theta_3 = 0.006$	$\theta_1 = 0.999$ $\theta_2 = 0.998$ $\theta_3 = 0.998$	$\theta_1 = 0.04$ $\theta_2 = 0.06$ $\theta_3 = 0.55$	$\theta_1 = 4.8 \pm 2.5$ $\theta_2 = 4.9 \pm 2.5$ $\theta_3 = 5 \pm 2.5$	$4.49 \times 10^{-5} \pm 7 \times 10^{-4}$

Todas las tablas Q de las pruebas muestran la misma tendencia, por lo que todas pudieron encontrar la manera de caracterizar los 3 grados de libertad en un solo modelo (controlador), obteniendo un aprendizaje exitoso (véase Figura 32). Se observa que sus porcentajes de exploración de la tabla son variados $Q_1 = 14.72\,\%, Q_2 = 13.65\,\%, Q_3 = 13.56\,\%, Q_4 = 94.95\,\%, Q_5 = 94.95\,\%, Q_6 = 89.29\,\%$. Esto se puede asumir que cada modelo aprendió a su modo pero con una tendencia muy marcada y similar en cada caso.

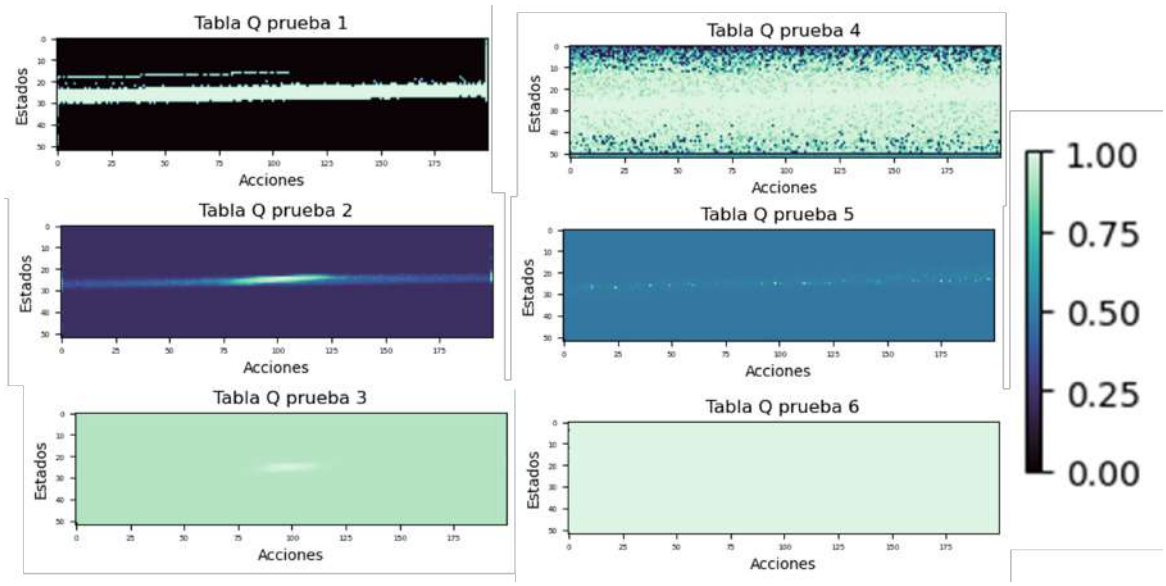


Figura 32: Valor Q normalizado de cada modelo. A la derecha con método UCB y a la izquierda con máxima recompensa (creación propia).

Las respuestas de los controladores fueron satisfactorias y son dinámicamente estables. Pues al realizar la prueba de varios pasos, continuaba realizando de manera correcta el movimiento parabólico como se muestra en la Figura 33. Comprobando que es posible que siga trayectorias a largo plazo. La salida se observa con un poco de ruido, pero esto se debe a la discretización de los estados y el número de iteraciones, por lo que al aumentar se mejorará la resolución de la salida.

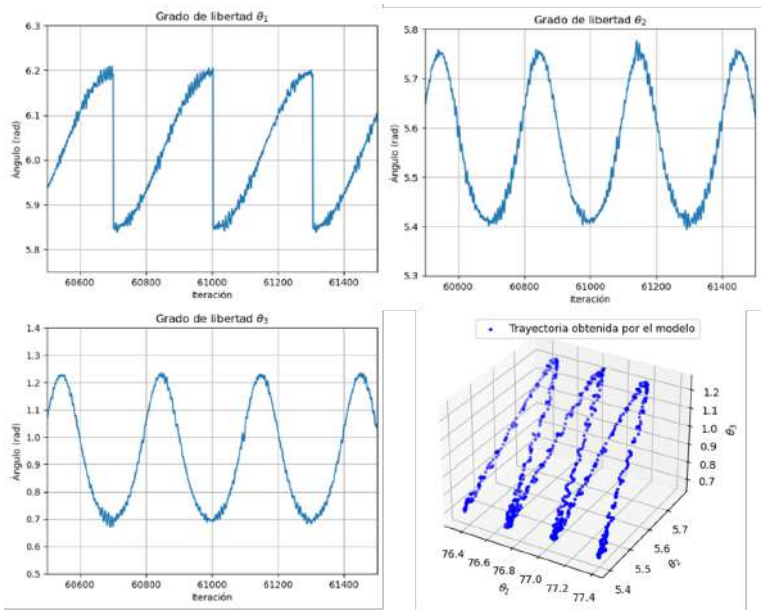


Figura 33: Trayectoria de pasos obtenidos por el modelo de la prueba 4 (creación propia).

Con los datos de salida de la respuesta de cada modelo, se realizó un análisis estadístico que permitió conocer más interpretaciones del modelo y sus resultados (véase Figura 34). Se observa que hay para el ángulo θ_1 se tienen 2 picos de frecuencia, los cuales son casi del mismo tamaño, siendo ligeramente más grande el de la izquierda. Esto significa que la mayoría de las pruebas inician con el estado inicial (pico izquierdo) y terminan casi todos en el estado final (pico derecho). En los ángulos $\theta_{2,3}$ se observa que el pico izquierdo resalta con respecto al derecho, esto se debe a que el estado inicial y el final es el mismo, por lo que la mayoría de las veces inicia y llega al mismo estado. Por otro lado, el pico derecho se ubica en el valor máximo del ángulo, lo que significa que las series de tiempo llegan al punto máximo (estado mayor) de la trayectoria Gaussiana.

Este análisis permitió conocer el comportamiento de las salidas de los modelos a través de varios pasos. Cabe resaltar que todos los controladores se observó el mismo comportamiento.

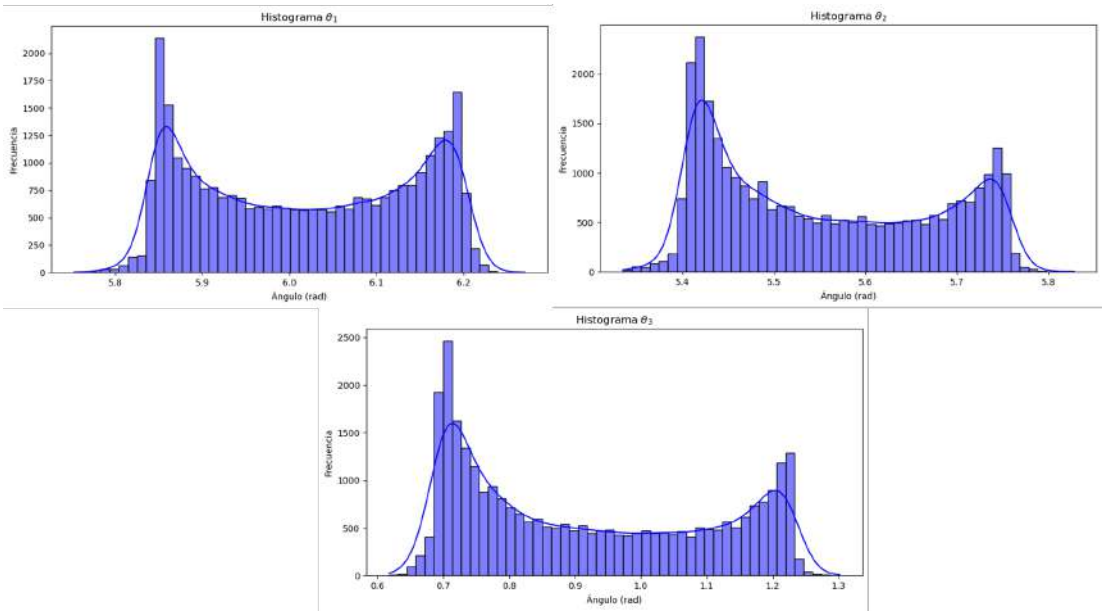


Figura 34: Histograma de las pruebas con el modelo de la prueba 4 (creación propia).

8.6. Validación de Resultados

Se realizaron comparaciones de las series de tiempo del modelo que mejor se consideró que trabajaba según lo dicho anteriormente (modelo 6), con el fin de poder generar hipótesis sobre el comportamiento de los mismos, fortaleciendo y complementando los resultados obtenidos. Obteniendo los siguientes resultados en la Tabla 6.

Tabla 6: Resultados de pruebas de hipótesis (creación propia).

Modelo (controlador)	Prueba de hipótesis	Valor de significancia	Resultado de la hipótesis nula
1	Cointegración	0.01	Las series de tiempo están cointegradas
2			Las series de tiempo están cointegradas
3			Las series de tiempo están cointegradas
4			Las series de tiempo están cointegradas
5			Las series de tiempo están cointegradas
6			Las series de tiempo están cointegradas
1	T de Student de dos muestras	0.01	Las series de tiempo no son significativamente diferentes
2			Las series de tiempo no son significativamente diferentes
3			Las series de tiempo no son significativamente diferentes
4			Las series de tiempo no son significativamente diferentes
5			Las series de tiempo no son significativamente diferentes
6			Las series de tiempo no son significativamente diferentes

En la prueba de cointegración se comprobó que las acciones futuras de los modelos seguirán la trayectoria de la serie de tiempo referida, debido a la comparación de sus residuales en la prueba de cointegración (Engle-Granger con coeficientes de MacKinnon). Asegurnado que al cambio de trayectoria (perturbaciones) podrá seguirlas el controlador respetando el umbral del residual a lo largo del tiempo (véase Figura 35 y 33).

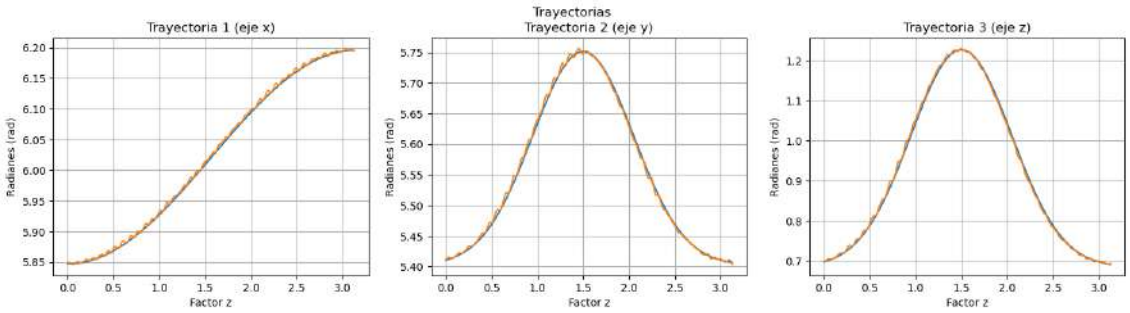
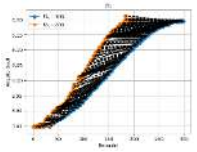
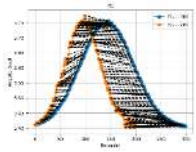
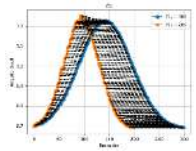
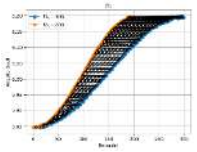
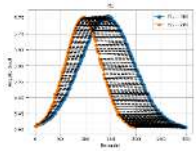
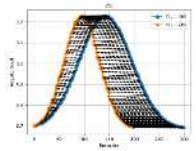
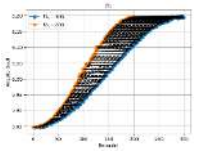
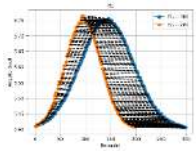
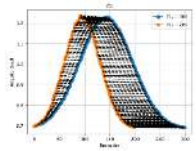
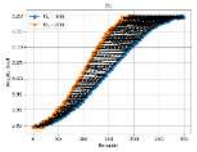
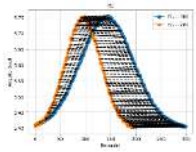
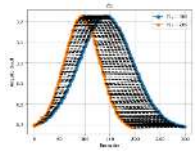
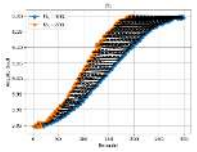
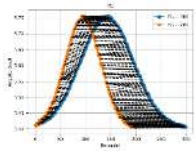
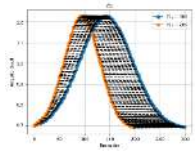
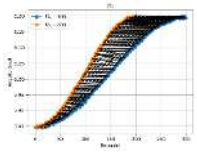
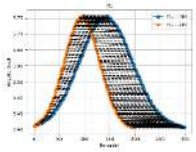
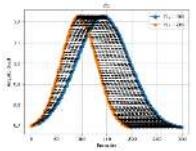


Figura 35: Salida del sistema con respecto a la trayectoria deseada (creación propia).

Adicionalmente, la prueba T de Student permitió realizar una comparación de las trayectorias deseadas con las salidas. Obteniendo un resultado de que no hay diferencias significativas entre las series de tiempo. Lo que indica que provienen de la misma distribución de datos y por lo tanto, comprueban que las métricas de evaluación son correctas y fiables.

Finalmente se realizó una comparación entre salidas de diferente longitud de la misma trayectoria, con el fin de observar la variación ante diferente cantidad de iteraciones. Para verificar y medir que tan diferente es un resultado de otro, se utilizará la distancia DTW. Se aplicó en cada grado de libertad con longitudes de las series de tiempo $t_{s1} = 300, t_{s2} = 200$ como se muestra en la Tabla 7.

Tabla 7: Resultados de pruebas de hipótesis (creación propia).

Modelo	$DTW(\theta_1)$	$DTW(\theta_2)$	$DTW(\theta_3)$
1	 8.2916	 2.521	 1.3446
2	 0.5323	 0.544	 0.7982
3	 0.5571	 0.6292	 0.7453
4	 2.06	 0.7563	 0.8696
5	 1.189	 1.485	 0.7449
6	 0.83	 0.49	 0.65

Se observó que las distancias son muy pequeñas, puesto que esta métrica tiene de rango $[0,\infty)$ y existe una buena alineación de los datos, pudiendo caracterizar un punto con más de uno del otro grupo. Por consiguiente, se puede decir que son resultados muy parecidos y por lo tanto, aunque incremente la cantidad de iteraciones, los resultados tendrán la misma consistencia.

9. Conclusiones

En este trabajo se presentó el desarrollo de un algoritmo basado en aprendizaje por refuerzo con políticas de aprendizaje Q y métodos exploratorios aplicado a un sistema de control dinámico que consistió en una pata robótica de 3 grados de libertad. Comparando las diferencias entre cada configuración del modelo controlador dentro de un ambiente de tareas episódicas en un desarrollo discretizado. Se desarrollaron 6 controladores que pudieron manejar el ambiente de una manera estable, mediante un entrenamiento basado en funciones de trayectorias con ruido aplicando transferencia de aprendizaje y ensamble. Las métricas de evaluación mostraron un buen control del sistema ($RMSE < 0.01$ para todos los modelos), incluso contra cambios en trayectorias. Además de que fue posible comprobar los resultados mediante pruebas de hipótesis, dando una interpretación más completa al trabajo y aportando información extra a los resultados.

El porcentaje de exploración de la tabla Q no es del todo determinante para el desempeño del modelo siempre y cuando se pueda generalizar el problema, utilizando una base de datos correcta y los hiperparámetros adecuados. También cabe resaltar que el diseño del ambiente es una pieza clave en el desarrollo del algoritmo, pues implica modificar el nivel de granularidad del sistema original con el que se está trabajando, ocasionando perdida en la información. Por lo que siempre debe de ser verificado.

La función $\varepsilon - Greedy$ permitió realizar tiempos de aprendizaje menores, a través de la estimación del valor ε, lo que ocasionó un balance entre explorar y explotar, permitiendo que el controlador pudiera explorar cuando sus métricas fueran en decrecimiento y explotar cuando sus métricas sean buenas.

La respuesta del sistema fue muy precisa y aunque se observó un poco de

ruido debido a la discretización del ambiente (por lo que si es requerido, se deberán agregar filtros para suavizar la salida del modelo), no es algo que afecte de una manera significativa el desempeño del modelo, siendo comprobado con la prueba de T de Student. También observó que aunque se modificara la cantidad de iteraciones t_s, se seguía realizando la tarea de manera episódica con éxito.

En el tiempo de ejecución y de entrenamiento, se observó que el método UCB es más veloz en comparación al método de máxima recompensa y aunque no tenga un porcentaje de exploración muy alto a largo plazo, tiene la capacidad de generalizar el problema. Siendo este más viable para un sistema embebido de recursos limitados al momento de implementarlo en un controlador digital.

Finalmente fue posible desarrollar pasos continuos (véase Figura 33), emulando el comportamiento de un insecto a través de las trayectorias deseadas. Logrando el objetivo planteado y realizando una aportación al desarrollo de trabajos relacionados con el control de sistemas mediante el uso de inteligencia artificial.

10. Aportación y Trabajos Futuros

- Se desarrolló una generalización de las funciones de trayectoria $\theta_{2,3}$ mediante funciones recursivas Gaussianas.
- Se construyó al menos un algoritmo de aprendizaje por refuerzo que pudiera controlar los 3 grados de libertad con una sola tabla Q.
- Se propuso una función $\varepsilon-Greedy$ para solucionar el dilema de exploración vs explotación, funcionando de manera exitosa.

- Se propuso una metodología de entrenamiento basada en la generación de datos sintéticos, en especial para este caso donde los datos eran escasos.

Se propone realizar un análisis complementario del sistema de control y el controlador para conocer su estabilidad exacta (mediante análisis en frecuencia y pruebas de estabilidad). También se propone implementar el algoritmo en un microcontrolador para conocer las capacidades del algoritmo en dispositivos con recursos limitados. Finalmente se propone realizar un análisis de muestras mediante alguna técnica de inteligencia artificial para poder clasificar los conjuntos de datos que son de mayor provecho para el algoritmo.

Bibliografía

[1] G. Carbone and M. A. Laribi, "Recent trends on innovative robot designs and approaches," 2023.

[2] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019.

[3] W. Pedrycz, "From data to information granules: An environment of granular computing," in *2021 IEEE 20th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, pp. 2–2, 2021.

[4] F. Tedeschi and G. Carbone, "Design Issues for Hexapod Walking Robots," *Robotics*, vol. 3, no. 2, pp. 181–206, 2014.

[5] F. Reyes-Cortés, *Robótica Control de robots manipuladores*. AlfaOmega, 2011.

[6] J. Coelho, F. Ribeiro, B. Dias, G. Lopes, and P. Flores, "Trends in the control of hexapod robots: a survey," *Robotics*, vol. 10, no. 3, p. 100, 2021.

[7] X. Ding and F. Yang, "Study on hexapod robot manipulation using legs," *Robotica*, vol. 34, no. 2, pp. 468–481, 2016.

[8] M. Laparm, *Deep Reinforcement Learning Hands-On*. Packt, 2018.

[9] M. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[10] J. Torres, *Introducción al aprendizaje por refuerzo profundo. Teoría y práctica en Python*. Kindle Direct Publishing, 2021.

[11] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer, "The Dynamics of Legged Locomotion: Models, Analyses, and Challenges," *SIAM Review*, vol. 48, no. 2, pp. 207–304, 2006. Publisher: Society for Industrial and Applied Mathematics.

[12] K.-Y. Tseng and P.-C. Lin, "A model-based strategy for quadruped running with differentiated fore- and hind-leg morphologies," *Bioinspiration and Biomimetics*, vol. 17, p. 026008, Feb. 2022. Publisher: IOP Publishing.

[13] Y. Zheng, K. Xu, Y. Tian, and X. Ding, "Different manipulation mode analysis of a radial symmetrical hexapod robot with leg–arm integration," *Front. Mech. Eng.*, vol. 1, April 2022.

[14] R.-R. Afshar, Y. Zhang, J. Vanschoren, and U. Kaymak, "Automated reinforcement learning: An overview," *Cornell University*, pp. 1–47, January 2022.

[15] F. X. Govers, *Artificial Intelligence for Robotics*, vol. 1. Packt Birmingham-Mumbai, 2018.

[16] H. Mengjie, Z. Xingxing, X. Liguo, M. Ross, P. Song, , and W. Jinshun, "A review of reinforcement learning methodologies on control systems for building energy," *Working papers in transport, tourism, information technology and microdata analysis*, 2018.

[17] J. Bagnato, *Aprende Machine Learning*. ISBN españa, 2020. ISBN: 8409258161.

[18] M. Silva, "Aprendizaje por refuerzo: Introducción al mundo del rl." `https://medium.com/aprendizaje-por-refuerzo-introducci%C3%B3n-al-mundo-del/aprendizaje-por-refuerzo-introducci%C3%B3n-al-mundo-del-rl-1fcfbaa1c87`, 2019.

[19] A. H. Klopf, "Brain function and adaptive systems: A heterostatic theory," No. 133, Air Force Cambridge Research Laboratories, 1972.

[20] C. Watkins, "Learning from delayed rewards," *King ´s College*, 1989.

[21] C. Watkins and P. Dayan, "Technical note: Q-learning," *Kluwer Academic Publishers*, vol. 9, pp. 279–292, 1992.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 3. The MIT Press, 1998.

[23] F.-Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang, and L. Yang, "Where does alphago go: From church-turing thesis to alphago thesis and beyond," *IEEE/CAA JOURNAL OF AUTOMATICA SINICA*, vol. 3, no. 2, pp. 113–120, 2016.

[24] M. Laparm, *Deep Reinforcement Learning Hands-On*. Packt, 2018.

[25] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "Experience selection in deep reinforcement learning for control," *Journal of Machine Learning Research*, vol. 19, pp. 1–56, 2018.

[26] T. T. Nguyen, N. D. Nguyen, P. Vamplew, S. Nahavandi, R. Dazeley, and C. P. Lim, "A multi-objective deep reinforcement learning framework," *The International Journal of Intelligent Real-Time Automation*, vol. 96, no. 103915, 2020.

[27] J. Dornheim, N. Link, and P. Gumbsch, "Model-free adaptive optimal control of episodic fixed-horizon manufacturing processes using reinforcement learning," *International Journal of Control, Automation and Systems*, vol. 18, pp. 1–12, 2019.

[28] A. S. Polydoros, L. Nalpantidis, and V. Krüger, "Advantages and limitations of reservoir computing on model learning for robot control," *Sustainable and Reliable Robotics for Part Handling in Manufacturing Automation (STAMINA)*, 2015.

[29] R. Arranz, L. C. Echeverría, J. R. D. Caño, F. Ponce, and J. L. Romero, "Aprendizaje por refuerzo profundo aplicado a juegos sencillos," *Universidad Complutense de Madrid*, 2019.

[30] Z. Li, X. Wei, X. Jiang, and Y. Pang, "A kind of reinforcement learning to improve genetic algorithm for multiagent task scheduling," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–12, 2021.

[31] A. Mullapudi, M. J. Lewis, C. L. Gruden, and B. Kerkez, "Deep reinforcement learning for the real time control of stormwater systems," *Advances in Water Resources*, vol. 140, no. 103600, 2020.

[32] G. Puriel-Gil, W. Yu, and H. Sossa, "Reinforcement learning compensation based pd control for inverted pendulum," *International Conference on Electrical Engineering, Computing Science and Automatic Control*, pp. 1–7, 2018.

[33] G. Puriel-Gil, W. Yu, and H. Sossa, "Reinforcement learning compensation based pd control for a double inverted pendulum," *IEEE Latin America Transactions*, vol. 17, no. 2, pp. 323–329, 2019.

[34] P. Klink, H. Abdulsamad, B. Belousov, C. D'Eramo, J. Peters, and J. Pajarinen, "A probabilistic interpretation of self-paced learning with applications to reinforcement learning," *Journal of Machine Learning Research*, vol. 22, pp. 1–52, 2021.

[35] J. E. Sierra-García and M.Santos, "Redes neuronales y aprendizaje por refuerzo en el control de turbinas eólicas," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 18, pp. 327–335, 2021.

[36] Z. Kegenbekov and I. Jackson, "Adaptive supply chain: Demand–supply synchronization using deep reinforcement learning," *Multidisciplinary Digital Publishing Institute (MDPI)*, vol. 14, no. 240, pp. 1–14, 2021.

[37] P. Vassiliadis, G. Derosiere, C. Dubuc, A. Lete, F. Crevecoeur, F. C. Hummel, and J. Duque, "Reward boosts reinforcement-based motor learning," *iScience*, vol. 24, no. 102821, 2021.

[38] S. Xu, I. Koren, and C. M. Krishna, "Adaptive workload adjustment for cyber-physical systems using deep reinforcement learning," *Sustainable Computing: Informatics and Systems*, vol. 30, no. 100525, 2021.

[39] H. Fu, K. Tang, P. Li, W. Zhang, X. Wang, G. Deng, T. Wang, and C. Chen, "Deep reinforcement learning for multi-contact motion plan-

ning of hexapod robots," *International Joint Conference on Artificial Intelligence (IJCAI-21)*, pp. 2381–2388, August 2021.

[40] T.-H. S. Li, P.-H. Kuo, L.-H. Chen, C.-C. Hung, P.-C. Luan, H.-P. Hsu, C.-H. Chang, Y.-T. Hsieh, and W.-H. Lin, "Fuzzy double deep q-network-based gait pattern controller for humanoid robots," *IEEE TRANSACTIONS ON FUZZY SYSTEMS*, vol. 30, pp. 147–161, January 2022.

[41] J. Torres, *Introducción al aprendizaje por refuerzo profundo. Teoría y práctica en Python*. Kindle Direct Publishing, 2021.

[42] M.-A. Chadi and H. Mousannif, "Understanding reinforcement learning algorithms: The progress from basic q-learning to proximal policy optimization," 2023.

[43] Y. Li, "Deep reinforcement learning: An overview," *Cornell University*, 2018.

[44] L. Rincón, *Introducción a los procesos estocásticos*. Departamento de Matemáticas UNAM, 2012.

[45] C. Acevedo, "Aplicación de cadenas de markov para el analisis y pronostico de series de tiempo." `http://tangara.uis.edu.co/biblioweb/tesis/2011/141227.pdf`, 2011. (Accessed on 03/02/2023).

[46] J.-M. Pastor, H. Díaz, L. Armesto, and A. Sala, "Aprendizaje por refuerzo con búsqueda de políticas: Simulación y aplicación a un sistema electromecánico," *Actas de las XXXVII Jornadas de Automática*, pp. 710–717, 2016.

[47] L. Buşoniu, R. Babuška, B. D. Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*, vol. 2. CRC Press, 2010.

[48] S. C. . L. A. almár, Z., "Module-based reinforcement learning: Experiments with a real robot.," *University of Alberta*, vol. 31, no. 3, pp. 55–85, 2018.

[49] A.-m. Farahmand and C. Szepesvári, "Model selection in reinforcement learning," *Machine learning*, vol. 85, no. 3, pp. 299–332, 2011.

[50] H. Jonathan, "Rl-actor-critic methods:a3c,gae,ddpg,q-prop." `https://jonathan-hui.medium.com/rl-actor-critic-methods-a3c-gae-ddpg-q-prop-e1c41f268541`, Mayo 2021.

[51] S. R. Eddy, "What is dynamic programming?," *Nature biotechnology*, vol. 22, no. 7, pp. 909–910, 2004.

[52] F. J. Beutler, "Dynamic programming: Deterministic and stochastic models," *SIAM Review*, vol. 31, no. 1, p. 132, 1989.

[53] F. Idalia, "Programación dinámica." `https://www.ingenieria.unam.mx/sistemas/PDF/Avisos/Seminarios/SeminarioV/Sesion6_IdaliaFlores_20abr15.pdf`, Abril 2015.

[54] J. V. U. K. Reza-Refaei Afshar, Yingqian Zhang, "Automated Reinforcement Learning: An Overview," *Cornell University*, pp. 1–47, Jan. 2022.

[55] L. Dongmin, "Maximum entropy reinforcement learning, stochastic control," *Arvix*, August 2019.

[56] K. Sivamayil, E. Rajasekar, B. Aljafari, S. Nikolovski, S. Vairavasundaram, and I. Vairavasundaram, "A systematic study on reinforcement learning based applications," *Energies*, vol. 16, p. 1512, Feb 2023.

[57] K.-Y. Tseng and P.-C. Lin, "A model-based strategy for quadruped running with differentiated fore- and hind-leg morphologies," *Bioinspiration and Biomimetics*, vol. 17, p. 026008, feb 2022.

[58] E. Gorrostieta, E. Vargas, and A. Aguado, "A neuro pd control applied for free gait on a six legged robot.," *WSEAS TRANSACTIONS ON COMPUTERS*, vol. 3, no. 4, pp. 1–7, 2004.

[59] M. García-López, E. Gorrostieta-Hurtado, J. R.-A. Emilio Vargas-Soto, A. Sotomayor-Olmedo, and J. Moya-Morales, "Kinematic analysis for trajectory generation in one leg of a hexapod robot.," *Iberoamerican Conference on Electronics Engineering and Computer Science*, vol. 3, pp. 342–350, 2012.

[60] A. Cervantes, E. Gorrostieta, J.-M. Ramos, C.-M. Torres, and E. Rivas, "Sistema de control pid sintonizado por algoritmo de evolución diferencial," *Asociación Mexicana de Mecatrónica, La mecatrónica en México*, vol. 10, pp. 47–60, Mayo 2021.

[61] V.M. Hernández-Guzmán and R. Silva-Ortigoza and R.V. Carrillo Serrano, *Control Automático: Teoría de diseño, Construcción de Prototipos, Modelado, Identificación y Pruebas Experimentales*. Colección CIDETEC del Instituto Politécnico Nacional, 2013.

[62] V.M. Hernández-Guzmán and R. Silva-Ortigoza, *Automatic Control with Experiments*. Springer, 2019.

[63] B. T, "Advanced time series analysis in python: Decomposition, autocorrelation." `https://towardsdatascience.com/advanced-time-series-analysis-in-python-decomposition-autocorrelation-115aa64f475e`, July 2021.

[64] IBM, "Funciones de autocorrelación y autocorrelación parcial." https://www.ibm.com/docs/es/spss-modeler/saas?topic=data-autocorrelation-partial-autocorrelation-functions, Ago 2021.

[65] P. Białas, P. Korcyl, and T. Stebel, "Analysis of autocorrelation times in neural markov chain monte carlo simulations," *Phys. Rev. E*, vol. 107, p. 015303, Jan 2023.

[66] J. MacKinnon, "Critical values for cointegration tests," *Long-run economic relationships*, vol. 13, 1991.

[67] J. G. MacKinnon, "Critical values for cointegration tests," tech. rep., Queen's Economics Department Working Paper, 2010.

[68] R. Montero, "Variables no estacionarias y cointegración." https://www.ugr.es/~montero/matematicas/cointegracion.pdf, Marzo 2013.

[69] E. Jaramillo and V. Lopez, "Tutorial para pruebas de cointegración de engle y granger en easyreg," *Departamento de Economía - Universidad Icesi*, Marzo 2011.

[70] A. Noriega and D. Ventosa-Santaulária, "Cointegración espuria: La prueba de engle-granger bajo la presencia de cambios estructurales," *Banco de México - Documentos de investigación*, Diciembre 2006.

[71] "Prueba t: Qué es, ventajas y pasos para realizarla." https://www.questionpro.com/blog/es/prueba-t-de-student/. (Accessed on 06/06/2023).

[72] "Prueba t de student proyecto papime unam pe." https://slideplayer.es/slide/16982894/. (Accessed on 06/06/2023).

[73] J. Zhang, "Dynamic time warping. explanation and code implementation." https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd, Feb 2020.

[74] M. Müller, *Information Retrieval for Music and Motion*. Springer Verlag, 2007.

[75] S. Roberts, "The upper confidence bound (ucb) bandit algorithm." https://towardsdatascience.com/the-upper-confidence-bound-ucb-bandit-algorithm-c05c2bf4c13f, Oct 2020.

[76] "Upper confidence bound algorithm in reinforcement learning." https://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/.

[77] C.-M. Aldo, G.-H. Efrén, R.-A. Juan-Manuel, and T. András, "Reinforcement learning applied to position control of a robotic leg: An overview," in *International Congress of Telematics and Computing*, pp. 335–351, Springer, 2022.

Anexos

Funciones de la Matriz de Torque

Sea la función de torque τ de 3 renglones (cada uno por cada grado de libertad), las siguientes expresiones de la función matricial desglosada que representa cada término $a_{o,p}, b_{o,p}, c_{o,p}, f_{o,p}$ en la ecuación (17).

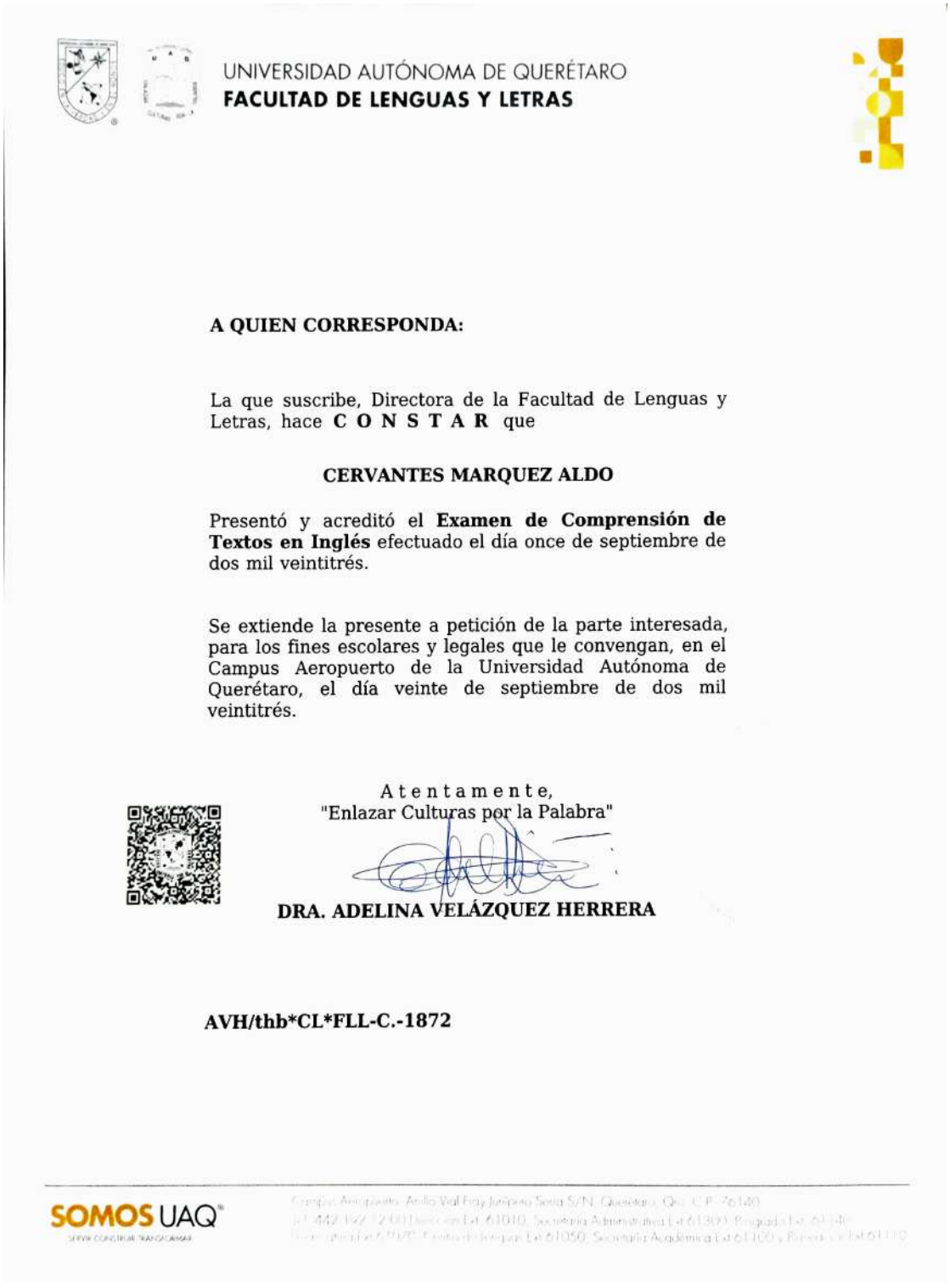
Donde J es la inercia, L es la longitud del eslabón y m es la masa del eslabón.

Tabla 8: Ecuaciones de la función de torque [58].

Parámetros de matriz de inercia	Parámetros de matriz Coriolis
$a_{11} = J_1 + L_2^2\cos^2(\theta_2) + \frac{1}{2}m_3L_2L_3\cos(\theta_2)\cos(\theta_3) + \frac{1}{4}m_3L_3^2\cos^2(\theta_3)$ $a_{22} = J_2 + \frac{1}{4}L_2m_2 + L_3m_3$ $a_{23} = \frac{1}{2}m_3L_2L_3(\sin(\theta_2)\sin(\theta_3) + \cos(\theta_2)\cos(\theta_3))$ $a_{32} = \frac{1}{2}m_3L_2L_3(\sin(\theta_2)\sin(\theta_3) + \cos(\theta_2)\cos(\theta_3))$ $a_{33} = \frac{1}{4}L_3m_3 + J_3$	$b_{21} = -\frac{1}{2}L_2^2(m_2\sin^2(\theta_2)\,m_3\sin^2(\theta_2)) - \frac{1}{4}m_3L_2L_3\cos(\theta_3)\cos(\theta_2)$ $b_{23} = \frac{1}{2}m_3L_2L_3(\sin(\theta_2)\cos(\theta_3) - \cos(\theta_2)\sin(\theta_3))$ $b_{31} = \frac{1}{2}m_3L_3\left(L_2\cos(\theta_2)\sin(\theta_3) - \frac{1}{2}\sin^2(\theta_3)\right)$ $b_{32} = \frac{1}{2}m_3L_2L_3(\sin(\theta_2)\sin(\theta_3) + \cos(\theta_2)\cos(\theta_3))$
Parámetros de matriz de centrifuga	**Parámetros de matriz de gravedad**
$c_{11} = -\left(m_3L_2^2\sin^2(\theta_2) + \frac{1}{2}m_3L_2L_3\sin(\theta_2)\cos(\theta_3) + \frac{1}{4}m_2L_2^2\sin^2(\theta_2)\right)$ $c_{12} = -\left(\frac{1}{2}m_3L_3L_2\cos(\theta_2)\sin(\theta_3) + \frac{1}{2}m_3L_3^2\sin^2(\theta_3)\right)$ $c_{23} = \frac{1}{2}m_3L_2L_3(\cos(\theta_2)\sin(\theta_3) - \sin(\theta_3)\cos(\theta_3))$ $c_{33} = \frac{1}{2}m_3L_2L_3(\sin(\theta_2)\cos(\theta_3) - \cos(\theta_2)\sin(\theta_3))$	$f_{22} = \frac{1}{2}L_2\cos(\theta_2)$ $f_{23} = L_2\cos(\theta_2)$ $f_{33} = \frac{1}{2}L_3\cos(\theta_3)$

Publicaciones y ponencias

Requisitos de idioma FLL



UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE LENGUAS Y LETRAS

A QUIEN CORRESPONDA:

La que suscribe, Directora de la Facultad de Lenguas y Letras, hace **C O N S T A R** que

CERVANTES MARQUEZ ALDO

Presentó y acreditó el **Examen de Comprensión de Textos en Inglés** efectuado el día once de septiembre de dos mil veintitrés.

Se extiende la presente a petición de la parte interesada, para los fines escolares y legales que le convengan, en el Campus Aeropuerto de la Universidad Autónoma de Querétaro, el día veinte de septiembre de dos mil veintitrés.

A t e n t a m e n t e,
"Enlazar Culturas por la Palabra"

DRA. ADELINA VELÁZQUEZ HERRERA

AVH/thb*CL*FLL-C.-1872

SOMOS UAQ

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE LENGUAS Y LETRAS

A QUIEN CORRESPONDA:

La que suscribe, Directora de la Facultad de Lenguas y Letras, hace **C O N S T A R** que

CERVANTES MARQUEZ ALDO

Presentó el **Examen de Manejo de la Lengua** efectuado el día nueve de febrero de dos mil veintitrés, en el cual obtuvo la siguiente calificación:

8+

Se extiende la presente a petición de la parte interesada, para los fines escolares y legales que le convengan, en el Campus Aeropuerto de la Universidad Autónoma de Querétaro, el día veintiuno de febrero de dos mil veintitrés.

A t e n t a m e n t e,
"Enlazar Culturas por la Palabra"

DRA. ADELINA VELÁZQUEZ HERRERA

AVH/daa*CL*FLL-C.-445