**Universidad Nacional Mayor de San Marcos**

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMATICA

****

**CURSO:** Automatización y Control de Software

**TEMA:** Péndulo Invertido Con Control Inteligente

**PROFESOR(A):** Yessica Rosas Cuevas

**INTEGRANTES:** GRUPO 4

* Ramirez Gomero Bryan Anthony
* Paredes Inga Bryan Gerardo
* Palomino Julian Alex Marcelo
* Cespedes Flores Sebastián
* Hinostroza De La Peña Cristhian
* Ancaya Martinez Benjamin

**Lima, Perú**

**2024**

[1. Introducción 3](#_3lzys31p6riu)

[2. Marco Teórico 3](#_atue69bctfgy)

[2.1. El péndulo invertido 3](#_rlwdxgia6bx5)

[2.2. Aprendizaje por refuerzo clásico: Q-learning 5](#_d52amwwuygrp)

[2.3. Entorno CartPole-v1 de Gym 6](#_j3vfdq6v78x)

[2.4. Función de recompensa 7](#_prgkxrbbawzn)

[2.5. Parámetros del Q-learning 7](#_iwzyrak19w3t)

[3. Implementación 7](#_b36ql1g8aj3l)

[3.1. Modelo del Sistema (CartPole) 8](#_xjlw4iyfrki)

[3.2. Agente Inteligente ( Agent y QLearning) 8](#_b97mgf36izg2)

[3.2.1. Agent: 8](#_ve8yg0m8a668)

[3.2.2. QLearning: 9](#_dlcdmyp6dtj)

[3.3. Entrenamiento y Prueba del Sistema 9](#_sz7wf2cwzw3)

[3.3.1. Entorno de Simulación (CartPole.py) 9](#_rrcit3cys66o)

[3.3.2. Algoritmo de Aprendizaje (Qlearning.py) 10](#_m4p4acda72fe)

[3.3.3. Lógica de Aprendizaje (learning.py) 11](#_kj36dpuowgs2)

[3.3.4. Evaluación del Sistema (test.py) 12](#_rw6qtnjdcj7l)

[3.3.5. Coordinación del Sistema (main.py) 12](#_7cjrex6xekqi)

[3.4. Visualización y Almacenamiento de Resultados 13](#_kbain71i42bb)

[3.4.1. Sistema de visualización (visualization.py): 13](#_32j4rsa0lgvh)

[3.4.2. Sistema de Almacenamiento (storage.py): 16](#_14z3m7gtra9u)

[4. Análisis Gráfico y respuesta a las preguntas 18](#_cmu3efwoqmm)

[4.1. Impacto del ajuste en la tasa de exploración y explotación 18](#_vi8m4qekvu8v)

[4.2. ¿Cómo afecta la longitud del péndulo o la fuerza máxima aplicable en el carro a la capacidad del agente de estabilizar el sistema? 21](#_8du74bva5baf)

[4.3. ¿Cómo influye la estructura de la función de recompensa en el desempeño del agente? Por ejemplo, ¿qué efecto tiene aumentar o disminuir las penalizaciones por movimientos excesivos del carro o la caída del péndulo? 24](#_39rmsmolg16)

[4.4. ¿Cuánto tiempo tarda el agente en aprender una política estable? Analiza cómo la tasa de aprendizaje, el factor de descuento (γ) y la política de exploración afectan el tiempo de convergencia. 28](#_enc1souah3mm)

[4.5. ¿Cómo responde el agente a estados iniciales desfavorables, como un ángulo del péndulo cercano al límite de caída o a una posición extrema del carro? 31](#_dqo1ye71n66x)

[5. Conclusiones 34](#_9p4ci4sj68ku)

[6. Referencias 34](#_4b8bqvo9r5ap)

[7. Anexos 34](#_uufpwv5z8f5a)

# **Introducción**

El péndulo invertido es un sistema dinámico complejo que presenta inestabilidad inherente, constituyendo un desafío fundamental en estrategias de control con aplicaciones desde el equilibrio humano hasta la estabilización de cohetes. Tradicionalmente estudiado mediante métodos como PID y LQR, estos enfoques encuentran limitaciones al requerir modelos matemáticos precisos que no capturan completamente la dinámica del sistema.

El aprendizaje por refuerzo (RL) es una solución a esto, ya que permite a los sistemas aprender comportamientos óptimos mediante interacción directa con su entorno, evitando la necesidad de modelos explícitos. En el artículo de Safeea (2024) se habla más de esto, en él se nos cuenta que el RL es más adecuado para comprender el comportamiento dinámico de un agente en interacción que para establecer asignaciones estáticas entre variables.

En el presente informe se detalla un proyecto que busca estabilizar un péndulo invertido utilizando Q-learning en el entorno CartPole-v1 de OpenAI Gym. El objetivo es diseñar un agente capaz de aprender una política de control óptima, explorando desafíos como el diseño de funciones de recompensa, discretización de estados y configuración de parámetros de aprendizaje y apoyándonos de gráficos donde podemos visualizar el aprendizaje del sistema

# **Marco Teórico**

## **El péndulo invertido**

El péndulo invertido es un sistema dinámico clásico ampliamente utilizado para estudiar control y aprendizaje por refuerzo debido a su naturaleza inherentemente inestable. Consiste en un péndulo montado en un carro que puede moverse a lo largo de una pista horizontal.

**Ecuaciones de movimiento**

Las ecuaciones del sistema se derivan a partir de la segunda ley de Newton o el principio de Lagrange, considerando los siguientes parámetros:

: Masa del péndulo.

: Masa del carro.

: Longitud del péndulo desde el pivote hasta el centro de masa.

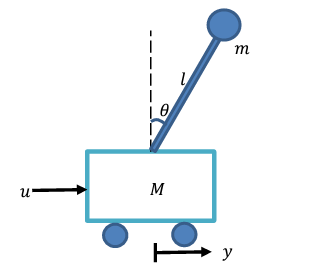
: Aceleración gravitatoria.

: Ángulo del péndulo respecto a la vertical.

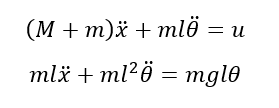
: Fuerza aplicada al carro.

**Figura 1.**

***Sistema de péndulo invertido, Yildiran, U. (2024)***



El modelo matemático del péndulo invertido puede describirse mediante las ecuaciones de movimiento derivadas del diagrama de cuerpo libre. Considerando un péndulo de longitud , mása , y un carro de mása , las ecuaciones que describen su dinámica se expresan como:



## **Aprendizaje por refuerzo clásico: Q-learning**

El aprendizaje por refuerzo (RL) es un paradigma de aprendizaje automático en el cual un agente aprende a tomar decisiones mediante la interacción con un entorno.

**Modelo de Aprendizaje por Refuerzo**

El problema de aprendizaje se modela como un Proceso de Decisión de Markov (MDP) definido por:

* **Espacio de estados ()**: Conjunto de posibles estados del sistema.
* **Espacio de acciones ()**: Conjunto de acciones que el agente puede realizar.
* **Función de transición ()**: Probabilidad de alcanzar el estado desde aplicando la acción .
* **Función de recompensa ()**: Valor recibido al realizar la acción en el estado .

**Algoritmo de Q-Learning**

El Q-Learning es un algoritmo off-policy que utiliza una tabla Q para aproximar la función de valor :



## **Entorno CartPole-v1 de Gym**

El entorno CartPole-v1 de OpenAI Gym simula un péndulo invertido sobre un carro, utilizado como entorno estándar para experimentos de aprendizaje por refuerzo.

**Descripción del Entorno**

**Espacio de estados**:  
Un vector continuo de 4 dimensiones:

* Posición del carro ().
* Velocidad del carro ().
* Ángulo del péndulo ().
* Velocidad angular del péndulo ().

**Espacio de acciones**:

* : Aplicar fuerza hacia la izquierda.
* : Aplicar fuerza hacia la derecha.

**Condiciones terminales**:

* El carro sale del rango permitido ().

**Recompensas**

El agente recibe una recompensa de +1 por cada paso que mantiene el péndulo en posición vertical, incentivando episodios más largos.

## **Función de recompensa**

La función de recompensa guía el comportamiento del agente al asignar un valor numérico a cada acción basada en su resultado. En el caso del CartPole, la recompensa estándar es:



## **Parámetros del Q-learning**

Los parámetros que influyen en el rendimiento del algoritmo son:

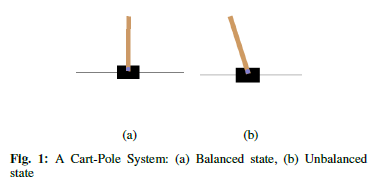
* **Tasa de aprendizaje ()**: Determina la rapidez con la que el agente adapta su conocimiento basado en nuevas observaciones.
* **Factor de descuento ()**: Define el peso de las recompensas futuras. Un valor cercano a 1 prioriza recompensas a largo plazo..

# **Implementación**

Usamos OpenAI Gym para simular el sistema CartPole, tal como se nos recomendó en clase y como recomienda Kumar (2020), que nos añade algunas estados del CartPole

**Figura 2.**

***Estados del Sistema CartPole de OpenAI Gym, Kumar, S. (2020)***



## **Entrenamiento y Prueba del Sistema**

El sistema desarrollado para el control del péndulo invertido implementa una arquitectura modular que permite la experimentación con diferentes configuraciones y parámetros. Esta flexibilidad facilita el análisis del comportamiento del sistema bajo diversas condiciones de entrenamiento y evaluación.

### **Componentes Base del Sistema**

El sistema se fundamenta en tres componentes principales que establecen la base para el aprendizaje y control:

**Entorno de Simulación (CartPole.py):**

La clase CarPole proporciona la conexión con el mundo físico simulado, extendiendo el entorno CartPole-v1 de Gymnasium. Este componente permite la configuración de parámetros físicos fundamentales:

* El ángulo inicial del péndulo (configurable en grados).
* El umbral de ángulo máximo permitido (valor base: 0.21 radianes).
* La longitud del péndulo (ajustable entre 0.5 y 1.0 unidades).
* La fuerza aplicable al carro (variable entre 1N y 10N).
* Los límites de posición del carro (±1.0 unidades).

El entorno implementa además un sistema de recompensas ponderado que evalúa:

* Verticalidad del péndulo (30% del peso): Utiliza una función coseno del ángulo para maximizar la recompensa cuando el péndulo está vertical (angle\_reward).
* Posición del carro (50% del peso): Recompensa la cercanía al centro de la pista mediante una función lineal basada en la distancia (position\_reward).
* Estabilidad del sistema (20% del peso): Considera las velocidades lineales y angulares para promover movimientos suaves (stability\_reward).

**Agente de Control (Agent.py):**

Este componente implementa la configuración de toma de decisiones del sistema, proporcionando:

* Discretización del espacio de estados continuo en 30 intervalos por dimensión.
* Política de selección de acciones ε-greedy con probabilidad configurable de exploración.
* Gestión de la tabla de valores Q mediante un diccionario de pares estado-acción.
* Mecanismo de balance entre exploración y explotación.

**Algoritmo de Aprendizaje (QLearning.py):**

Implementa el núcleo del aprendizaje por refuerzo mediante el algoritmo Q-learning:

* Tasa de aprendizaje (α) configurable para controlar la velocidad de actualización.
* Factor de descuento (γ) ajustable para balancear recompensas inmediatas y futuras.
* Mecanismo de actualización de valores Q basado en la ecuación de Bellman.
* Gestión de estados terminales y no terminales en el proceso de aprendizaje.

### **Lógica de Aprendizaje (learning.py)**

El módulo de aprendizaje implementa el ciclo de entrenamiento mediante la función train\_pendulum. Sus características principales incluyen:

Control de Entrenamiento:

* Número de episodios configurables (experimentalmente entre 2000 y 15000).
* Gestión de tasa de exploración variable (rangos estudiados: 20% a 50%).
* Implementación de decaimiento de exploración adaptativo.
* Registro de métricas durante el entrenamientos

### **Evaluación del Sistema (test.py)**

El sistema de evaluación implementa un protocolo de pruebas que permite analizar el rendimiento del agente bajo diferentes condiciones:

Protocolo Base de Evaluación:

* Ejecución de episodios de prueba (típicamente 10 episodios).
* Límite de 1000 pasos por episodio.
* Restricción temporal de 30 segundos por episodio.

Métricas de Evaluación:

* Tiempo de estabilización del péndulo.
* Desviación angular respecto a la vertical.
* Control de posición del carro.
* Acumulación de recompensas.

### **Coordinación del Sistema (main.py)**

El módulo principal coordina la interacción entre componentes y permite la configuración de experimentos con diferentes parámetros:

Gestión de Configuración:

* Inicialización de parámetros físicos del sistema.
* Configuración de hiper parámetros de aprendizaje.
* Control de condiciones iniciales de entrenamiento.

Control de Ejecución:

* Coordinación de fases de entrenamiento y evaluación
* Gestión de visualización de resultados
* Control de almacenamiento de datos

Esta arquitectura modular y configurable permite realizar los diversos experimentos presentados en la sección de análisis, donde se estudia el impacto de diferentes configuraciones en el rendimiento del sistema, incluyendo variaciones en la longitud del péndulo, fuerza aplicada, tasas de exploración y parámetros de aprendizaje.

## **Visualización y Almacenamiento de Resultados**

### **Sistema de visualización (visualization.py):**

El sistema de visualización lo implementamos a través de la clase VisualizationManager, que proporciona un conjunto comprehensivo de herramientas para el análisis visual de los resultados del entrenamiento y evaluación del agente. La implementación utiliza las bibliotecas matplotlib, seaborn y numpy para generar visualizaciones detalladas y estadísticamente significativas.

Componentes Principales:

* Visualización del Progreso de Entrenamiento

La función plot\_training\_progress implementa dos visualizaciones complementarias:

* + Un gráfico temporal que muestra la evolución de las recompensas por episodio, incluyendo tanto los valores individuales como un promedio móvil con una ventana de 100 episodios para suavizar las fluctuaciones.
  + Un histograma que representa la distribución de recompensas, permitiendo analizar la variabilidad del rendimiento del agente.
* Análisis Comparativo de Parámetros

Mediante plot\_parameter\_comparison, el sistema genera visualizaciones que permiten:

* + Comparar el desempeño del agente bajo diferentes configuraciones de parámetros.
  + Analizar la evolución del promedio de recompensas para cada configuración.
  + Identificar las configuraciones más efectivas para el aprendizaje.
* Análisis de Estados

La función plot\_state\_analysis genera un conjunto de visualizaciones que incluyen:

* + Distribución de posiciónes del carro.
  + Distribución de ángulos del péndulo.
  + Diagrama de fase que relaciona posición y velocidad.
  + Visualización de las acciones tomadas en diferentes estados.
* Análisis de Convergencia

A través de plot\_convergence\_analysis, el sistema proporciona:

* + Evolución temporal de las recompensas totales.
  + Progresión de la duración de los episodios.
  + Análisis bidimensional de la relación entre duración y recompensa.
* Generación de Reportes

La función create\_training\_report integra:

* + Estadísticas sumarias del entrenamiento.
  + Visualizaciones comprehensivas del desempeño.
  + Métricas de éxito y progreso.

Integración con el Sistema:

El sistema de visualización se integra con los módulos de entrenamiento y evaluación a través de:

* main.py: Visualiza la evolución de las recompensas durante el entrenamiento y los resultados de las pruebas mediante gráficos de series temporales.
* test.py: Genera visualizaciones específicas para la fase de evaluación, incluyendo:
  + Evolución del ángulo del péndulo.
  + Perfil de velocidad angular.
  + Recompensas acumuladas.
  + Tiempos de ejecución por episodio.

### **Sistema de Almacenamiento (storage.py):**

El sistema de almacenamiento se implementa mediante la clase ValueFunctionStorage, que proporciona funcionalidades para la persistencia y recuperación de la función de valor Q del agente. Este sistema opera de la siguiente manera:

Inicialización:

* La clase se inicializa con un nombre de archivo predeterminado "value\_function.pkl".
* Se utiliza para almacenar la tabla Q (value\_function) del agente de forma persistente.

Funcionalidades Principales:

* Almacenamiento (save):
  + Serializa la función de valor del agente utilizando pickle
  + Guarda los datos en un archivo binario con extensión .pkl
  + Confirma el guardado exitoso mediante un mensaje en consola
* Recuperación (load):
  + Intenta cargar una función de valor previamente guardada
  + Maneja casos de error cuando el archivo no existe
  + Inicializa un diccionario vacío si no se encuentra el archivo
  + Notifica el resultado de la operación mediante mensajes en consola.

Integración con el Sistema:

* El sistema de almacenamiento se integra en el flujo principal del programa a través de main.
* Se inicia al comienzo de la ejecución.
* Guarda automáticamente el estado final del agente después del entrenamiento.
* Permite la persistencia del conocimiento adquirido entre diferentes sesiones de entrenamiento.

# **Análisis Gráfico y respuesta a las preguntas**

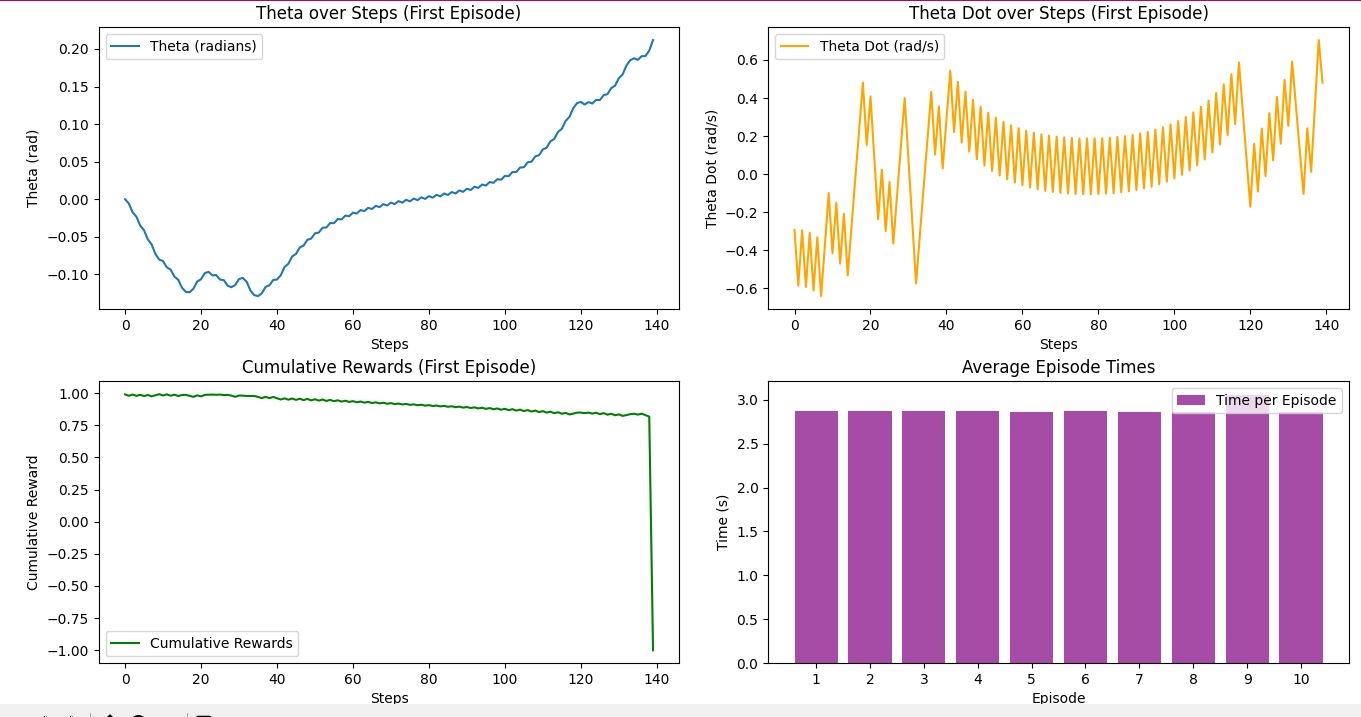
## **Impacto del ajuste en la tasa de exploración y explotación**

* **Entrenamiento de 2000 episodios con 20% de exploración y 80% de explotación.**

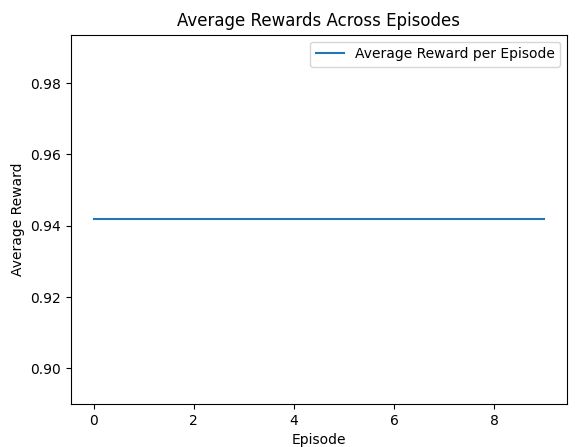
Se disminuyó la tasa de exploración a partir del episodio 1000 y con ello se observa el incremento de recompensa desde ese episodio. A partir del episodio 1000 hay estabilidad con mayores recompensar al explotar y maximizar una decisión a tomar.



En el test se observa un tiempo promedio de estabilidad de 2.8 segundos. El agente mantiene al péndulo en los límites por un pequeño periodo de tiempo antes de excederlo. Vemos que el ángulo del péndulo se incrementa y se abre con el tiempo lo que muestra que el carrito no logra aplicar el torque necesario para reducir ese ángulo.

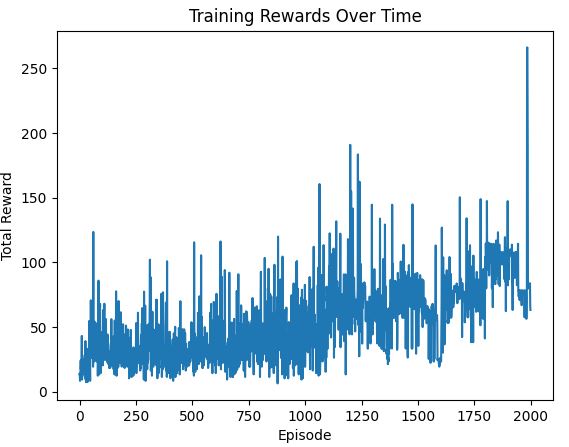


Finalmente la recompensa promedio en cada episodio es del 0.95 siendo la máxima recompensa 1 podemos mencionar que la política aprendida no es suficiente para estabilizar el péndulo. Incrementar el número de episodios permitirá al agente mejorar sus decisiones.

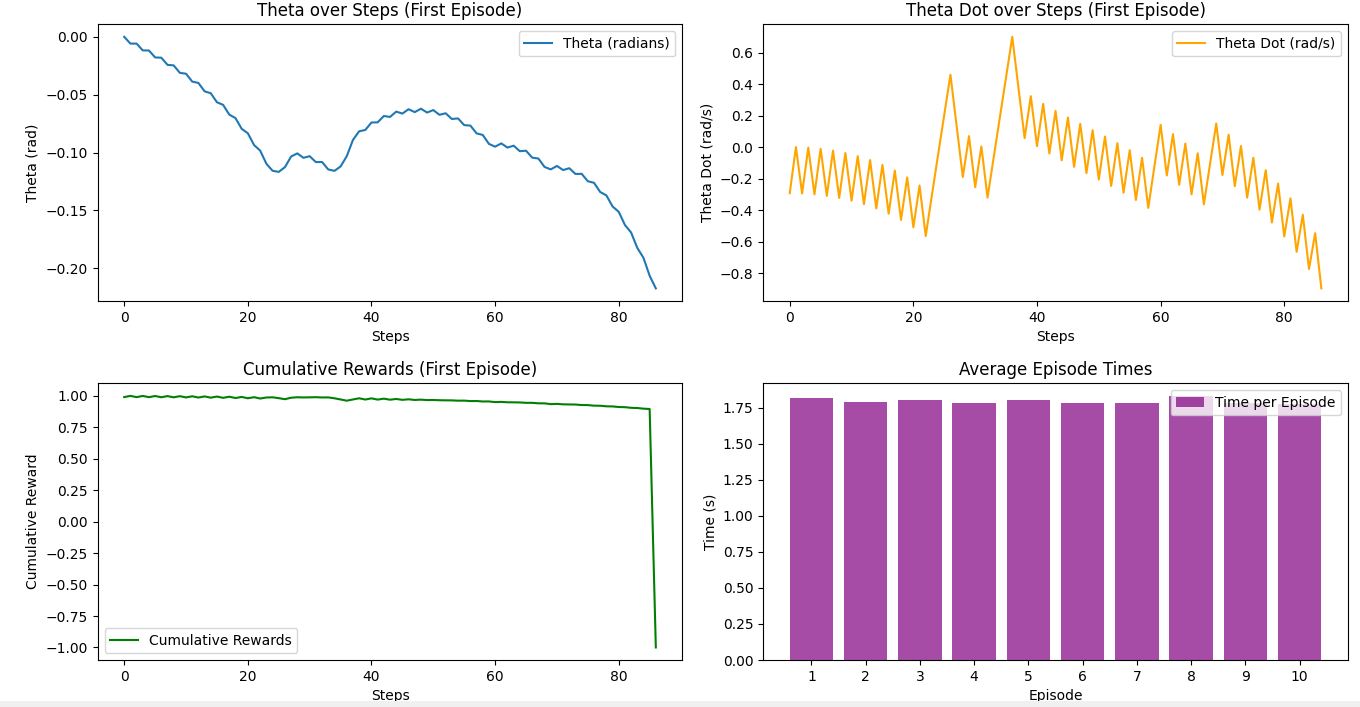


* **Entrenamiento de 2000 episodios con 50% de exploración y 50% de explotación.**

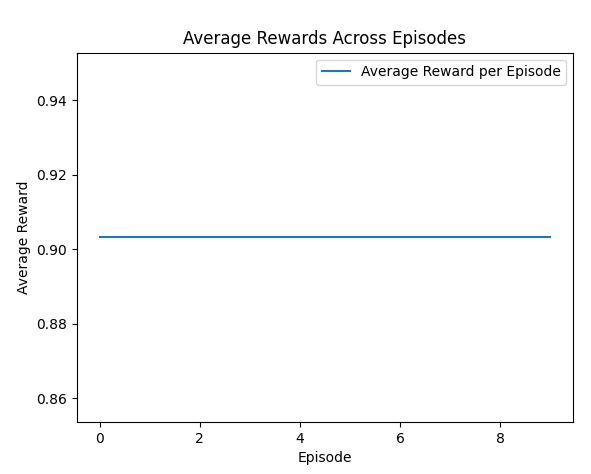
Se disminuyó la tasa de exploración a partir del episodio 1000 y con ello se observa el incremento de recompensa desde ese episodio. No hay una tendencia a incrementar la recompensa total desde el inicio debido a que se opta por explorar nuevas acciones. La explotación tampoco no muestra una tendencia rápida de mejora debido al poco número de episodios en los que explorara. El aumento de la tasa de explotación permite al agente afinar su decisiones y en este escenario el agente tiene insuficientes episodios para explotar.



En el test se observa un tiempo promedio de 1.75 inferior al tiempo promedio del escenario con tasa de exploración de 20%. También el ángulo se abre al lado izquierdo del límite vertical permitido debido a la insuficiente fuerza aplicada por el carrito.



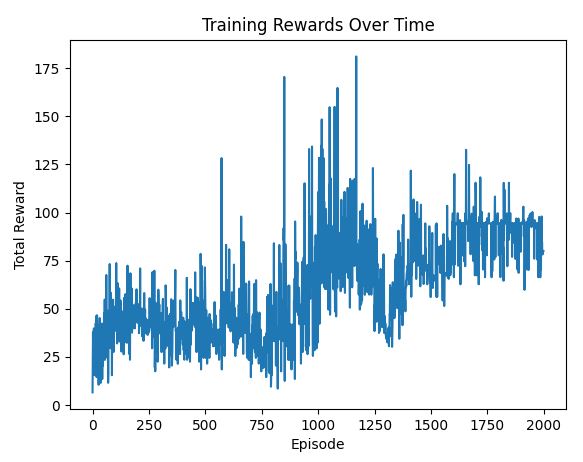
La recompensa promedio por episodio es menor que el primer escenario ya que permanece menor tiempo estable. Se aprecia que la tasa de exploración mejor es más efectiva para encontrar una política estable pero aun así no es la política que logra estabilizar el péndulo. Un mayor número de episodios podrá encontrar una política más estable permitiendo explotar más el aprendizaje del agente.



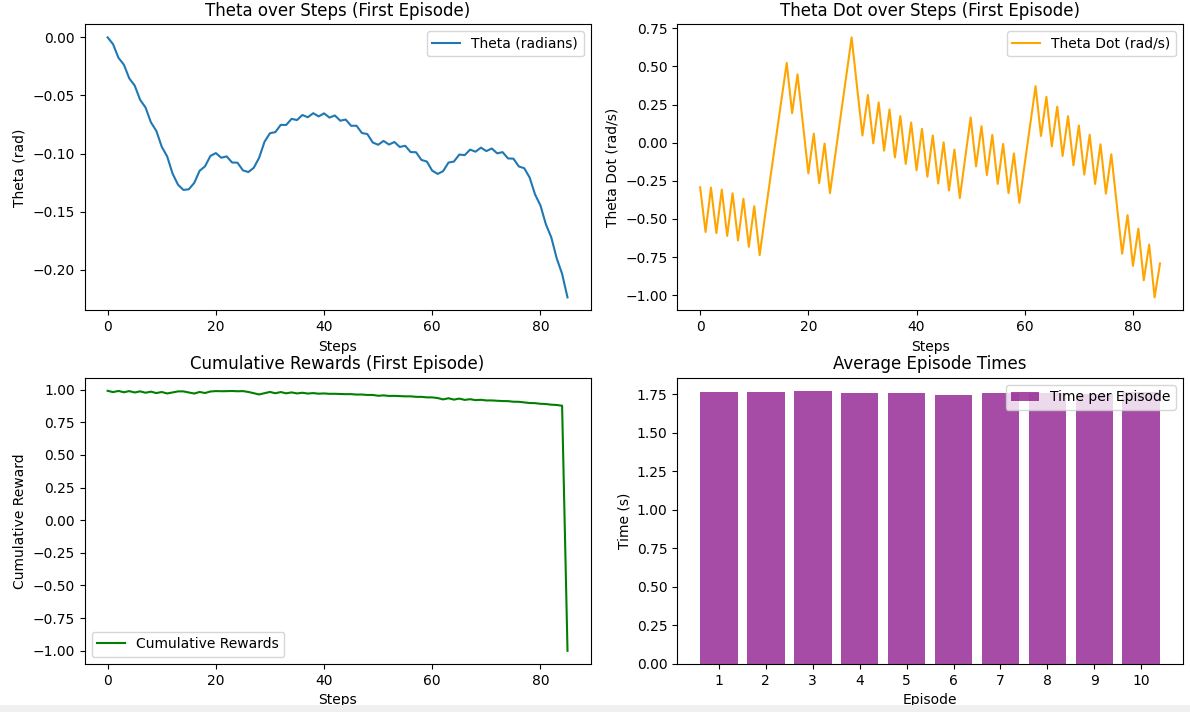
## **¿Cómo afecta la longitud del péndulo o la fuerza máxima aplicable en el carro a la capacidad del agente de estabilizar el sistema?**

* **Entrenamiento de 2000 episodios con 20% de exploración y una longitud de péndulo de 0.5m y una fuerza aplicada de 10N.**

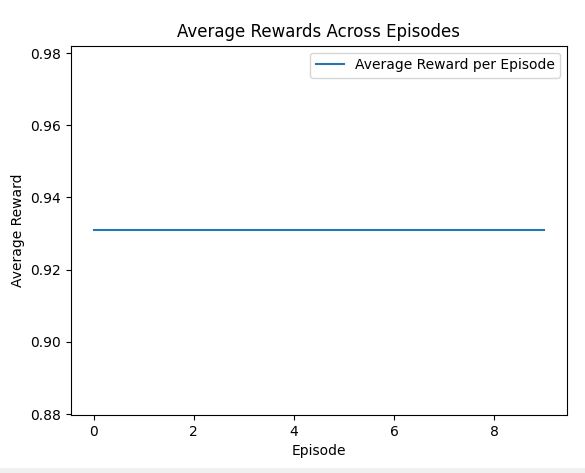
Observamos una estabilidad de la política a partir del episodio 1500.



Se observa en el entrenamiento que la política aprendida no es suficiente para mantener el péndulo en vertical, apreciamos que la fuerza del carrito no es suficiente debido a que no se ha establecido una política estable por el poco número de episodios de entrenamiento.

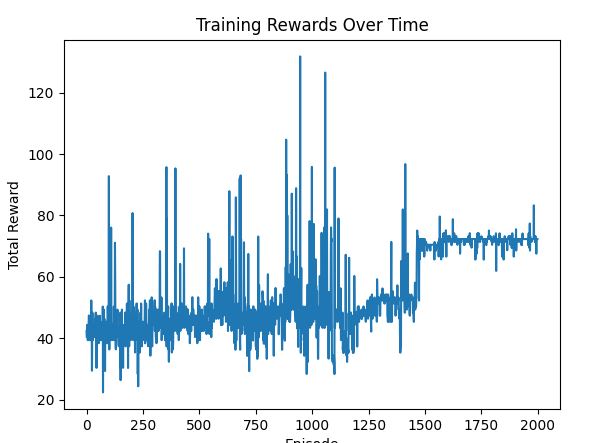


En el test se aprecia un tiempo de estabilidad de 1.75 segundos aproximadamente lo que evidencia la deficiencia de la política aprendida. Vemos una recompensa de 0.93 de 1 lo cual evidencia la falta de entrenamiento del agente.

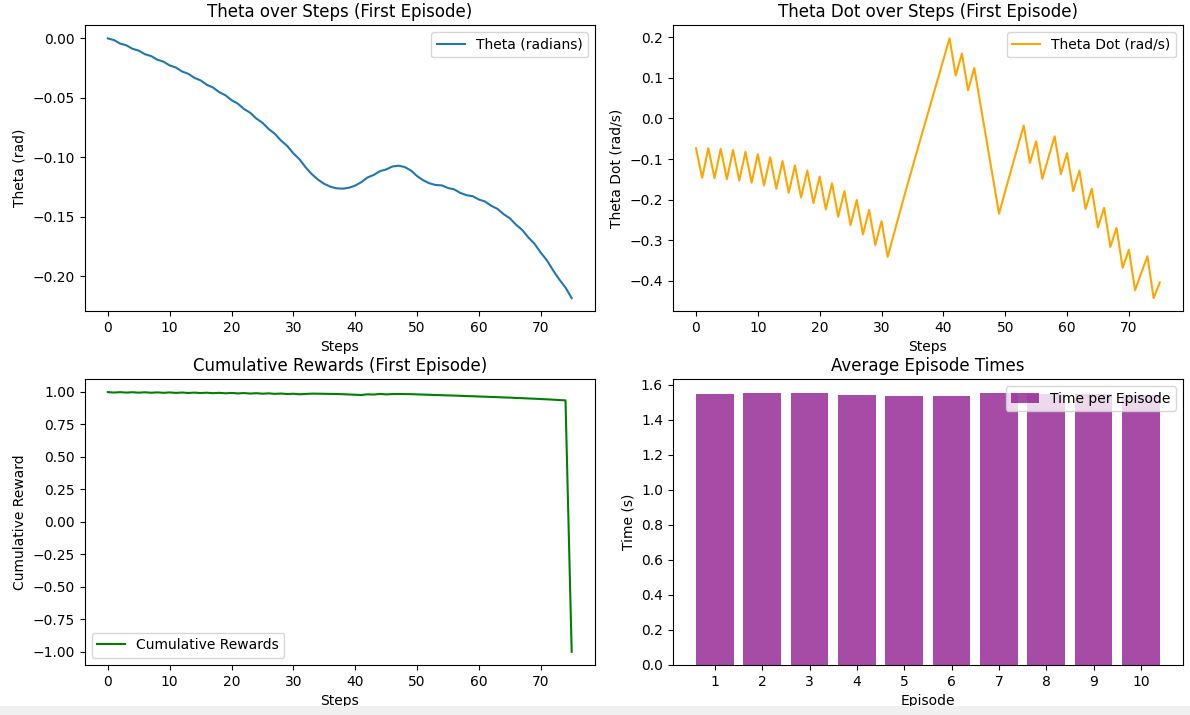


* **Entrenamiento de 2000 episodios con 20% de exploración y una longitud de péndulo de 1.0m y una fuerza aplicada de 5N.**

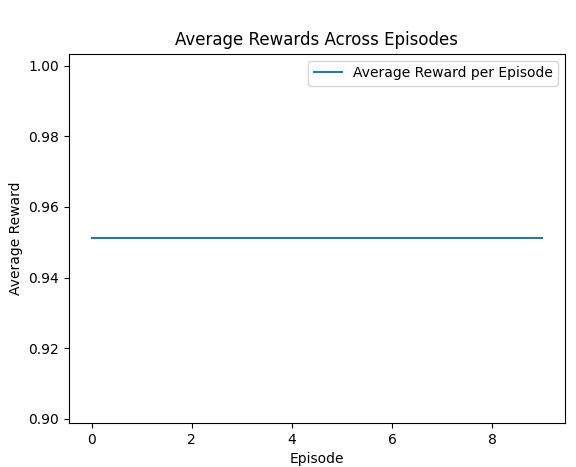
Ahora la longitud del péndulo se ha duplicado. Se aprecia una estabilidad de la política a partir del episodio 1500.



En el test apreciamos que el tiempo de estabilidad es menor en comparación al tiempo de estabilidad en la prueba del péndulo de 0.5m. Vemos un cambio suave y lento del ángulo del péndulo lo que evidencia un mayor control del péndulo mientras más grande es su longitud, esto debe a que la longitud incrementa el torque que aplique el carrito permitiendo estabilizarlo con menos aplicación de torque. La magnitud de la fuerza permite al carrito reaccionar con mayor rapidez y reaccionar para estabilizar el péndulo.



En el test el tiempo de estabilidad es de 1.6 y evidencia el insuficiente tiempo de entrenamiento del carrito. Aun así el aumento de la longitud del péndulo resulta en mejores resultados de recompensa desde el inicio del entrenamiento y desde la exploración.



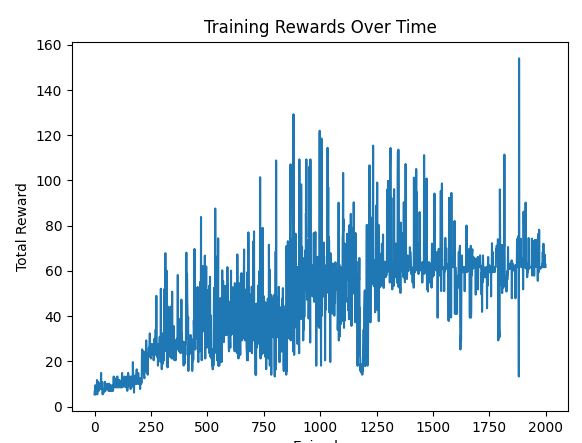
## **¿Cómo influye la estructura de la función de recompensa en el desempeño del agente? Por ejemplo, ¿qué efecto tiene aumentar o disminuir las penalizaciones por movimientos excesivos del carro o la caída del péndulo?**

* **Entrenamiento de 2000 episodios con 20% de exploración y una mayor recompensa por el movimiento excesivo.**

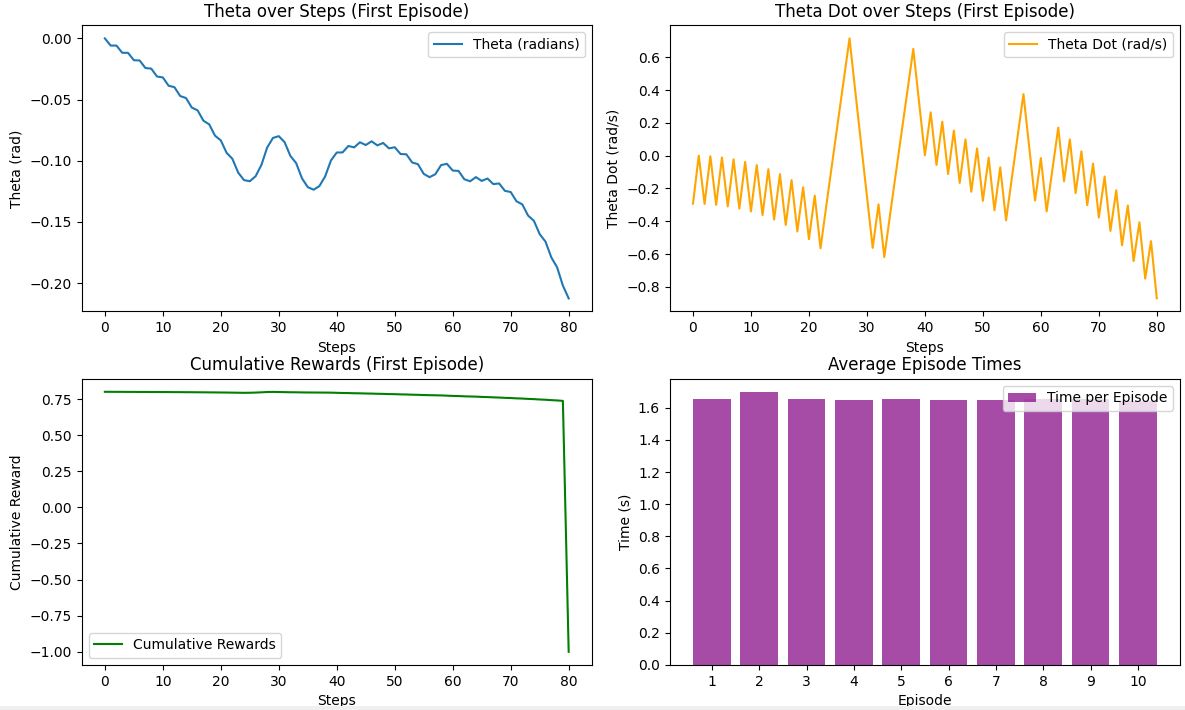
El movimiento excesivo se calculo es función de la posición anterior y la posición actual, además se normalizo a 1 para permear con 1 un movimiento excesivo.

lot\_of\_movement\_reward = abs(cart\_position - self.previous\_position) / 2

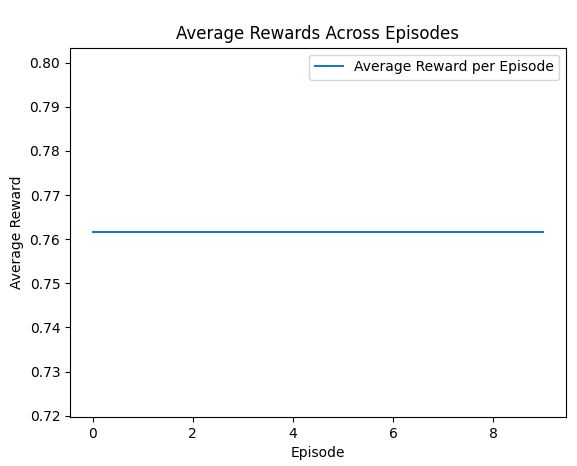
La política evidencia una tendencia a crecer rápidamente a partir del episodio 250, lo que significa que recompensar los movimientos excesivos permite mantener al ángulo en el límite vertical, esto en perjuicio de la estabilidad suave del péndulo.



En el test se observa un cambio turbulento en el ángulo del péndulo y de igual manera se aprecia el cambio brusco de la velocidad del péndulo. El tiempo de estabilización promedio es de 1.6 segundos y evidencia el poco número de episodios de entrenamiento.



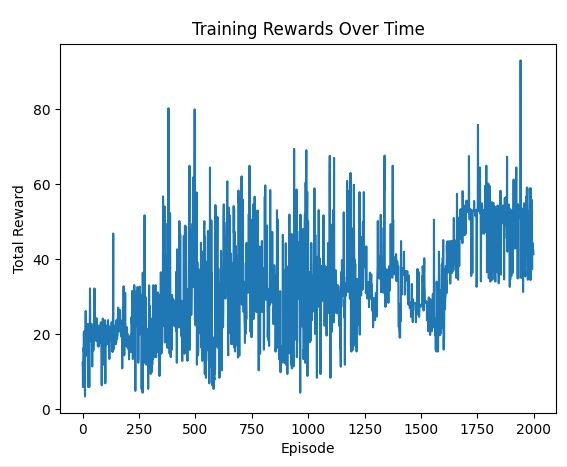
La recompensa promedio en cada episodio es de aproximadamente de 0.77 de 1 y ello se explica con la inestabilidad del péndulo por estar oscilando fuera de la posición vertical absoluta de 0 radiante.



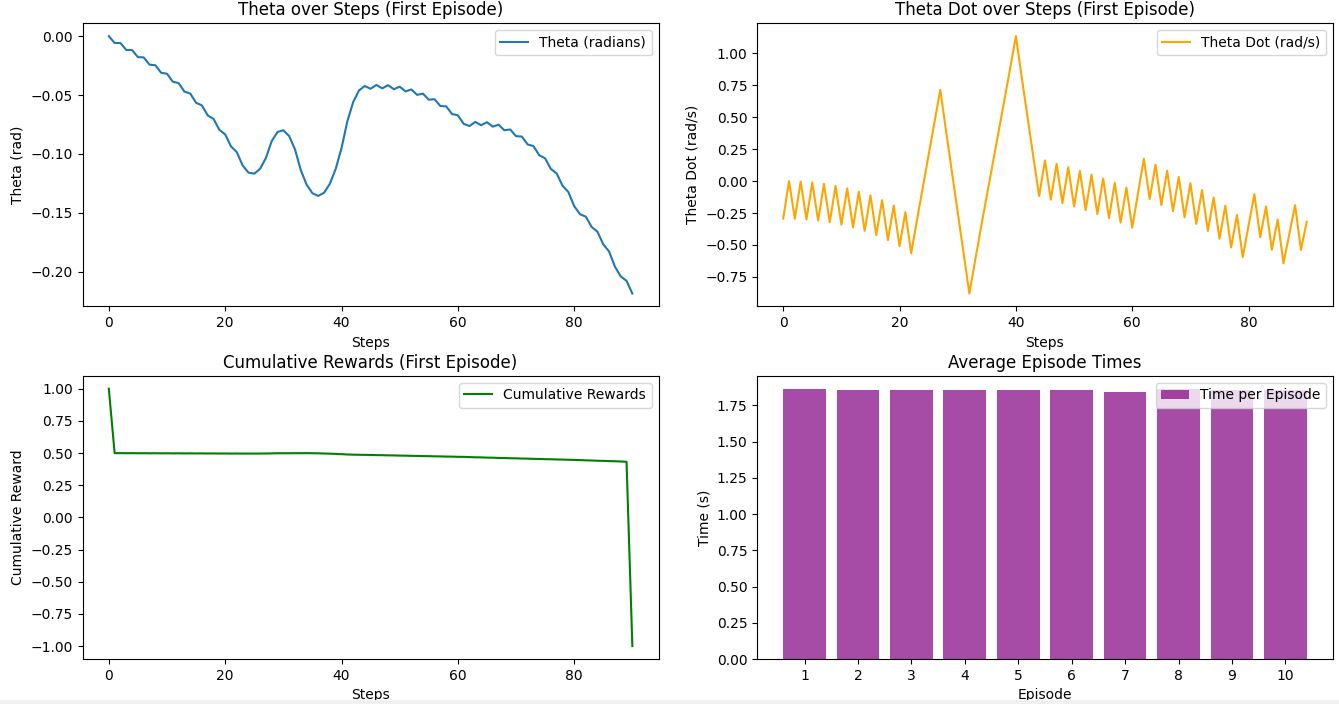
* **Entrenamiento de 2000 episodios con 20% de exploración y una mayor recompensa por el movimiento excesivo.**

Se recompensa el poco movimiento. little\_movement\_reward = 1 - lot\_of\_movement\_reward

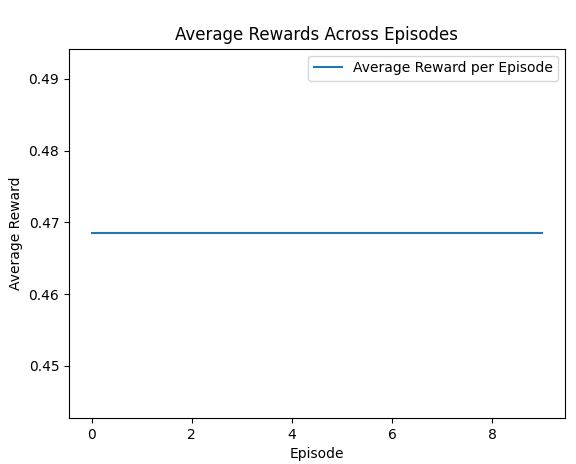
En el entrenamiento se observa un crecimiento lento de la recompensa total por episodio debido a que el carro se mueve lento y no lograr aplicar el torque necesario para lograr mantener el péndulo en vertical. Aunque a partir del episodio 1000 se incrementa la explotación el movimiento lento es perjudicial para lograr mantener el péndulo en vertical.



En el test el tiempo promedio de estabilización es superior al escenario anterior sin embargo la recompensa es muy inferior. El poco torque que se aplica con el poco movimiento resulta para el péndulo permanecer un mayor tiempo en límite del ángulo alejado de la posición vertical absoluta en donde la recompensa es mínima.



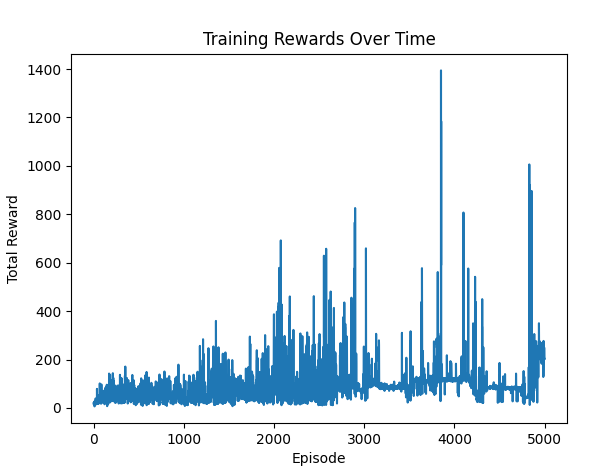
Finalmente la recompensa promedio por cada acción en cada episodio es 0.47 de 1 lo cual evidencia una pésima política aprendida. Un mayor número de episodios significa mejor toma de decisiones pero tomará más tiempo en lograrlo.



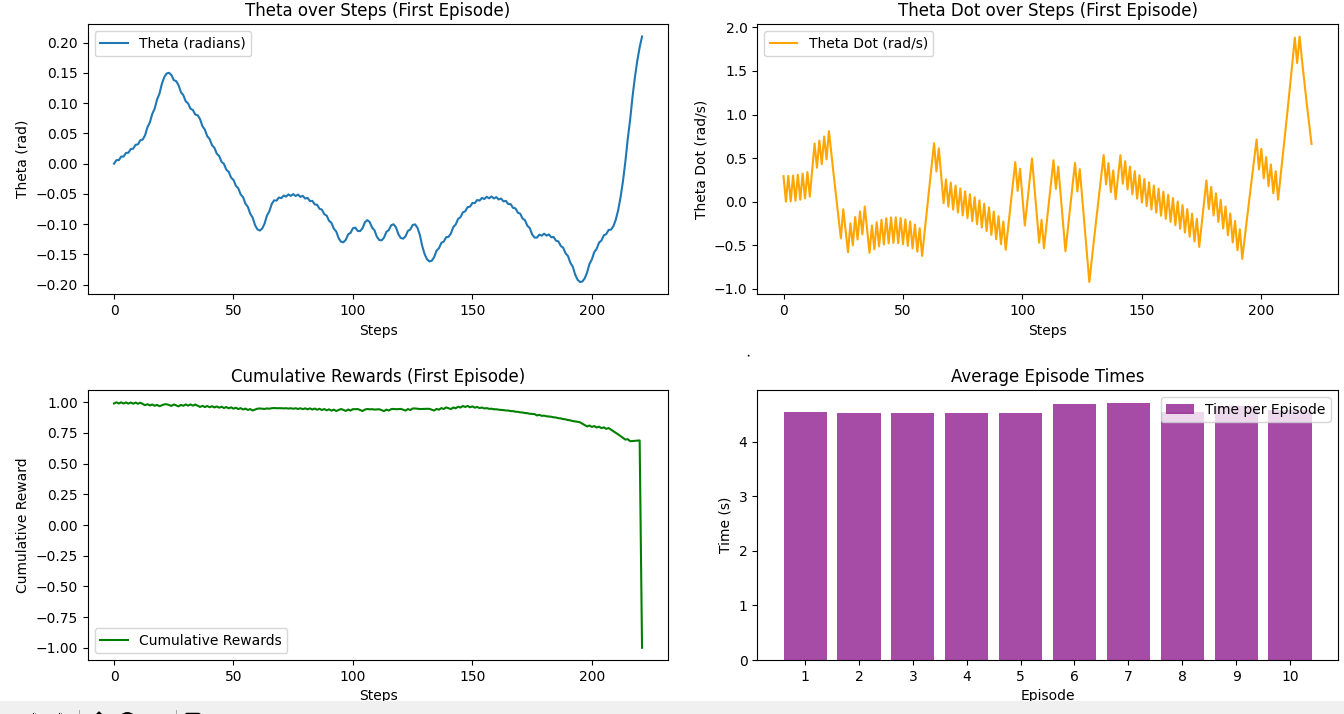
## **¿Cuánto tiempo tarda el agente en aprender una política estable? Analiza cómo la tasa de aprendizaje, el factor de descuento (γ) y la política de exploración afectan el tiempo de convergencia.**

* **Entrenamiento de 5000 episodios con 20% de exploración y alpha=0.2, gamma=0.80.**

Con un alpha de 0.2 el agente prioriza el aprendizaje rápido además con un gamma de 0.8 el agente pone más énfasis en las recompensas inmediatas. No vemos una tendencia de sostenible de mejora de la política porque el agente aprende rápido de las nuevas acciones lo que significa una modificación mayor con cada nuevo aprendizaje.

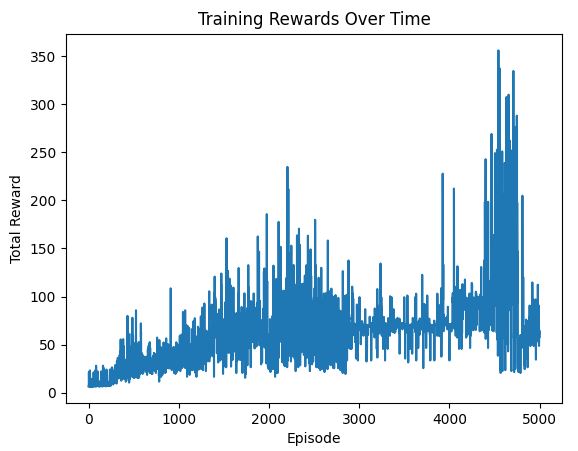


Observamos que el tiempo de estabilización es de 4 segundos. Debido a la poca fuerza del carro de 1N el agente logra evitar una sola vez que el péndulo alcance el límite de operación pero no es suficiente tener estabilidad sostenida.

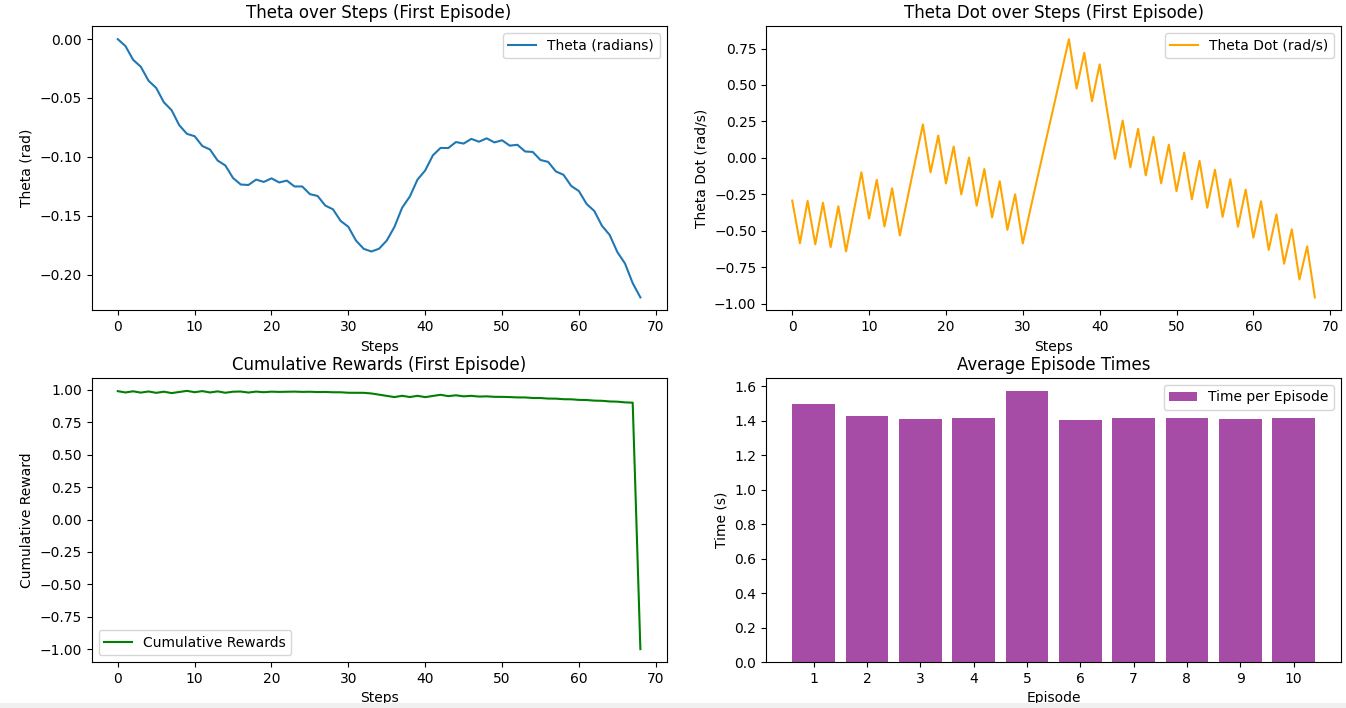


* **Entrenamiento de 5000 episodios con 20% de exploración y alpha=0.05, gamma=0.90.**

Con un alpha bajo la convergencia es superlento y el agente no aprenderá de las nuevas situaciones. Hay un aprendizaje lento que se evidencia en la baja recompensa durante gran parte del entrenamiento. La tasa gama de enfoque en recompensas futuras es grande lo que considera relevante las decisiones a largo plazo. Aun así la convergencia es lenta.

****

Vemos en el primer episodio del test que las recompensas para cada instante es cercano a 1 para los primeros pasos pero es insuficiente para mantener el péndulo en vertical. El tiempo de estabilidad es de 1.6 segundos y el ángulo del péndulo tiende a abrirse intentando sostenerse en el límite de operación. El carrito intenta mantener la estabilidad pero el torque de 1N y el aprendizaje es insuficiente para evitar que el se estabilice.

****

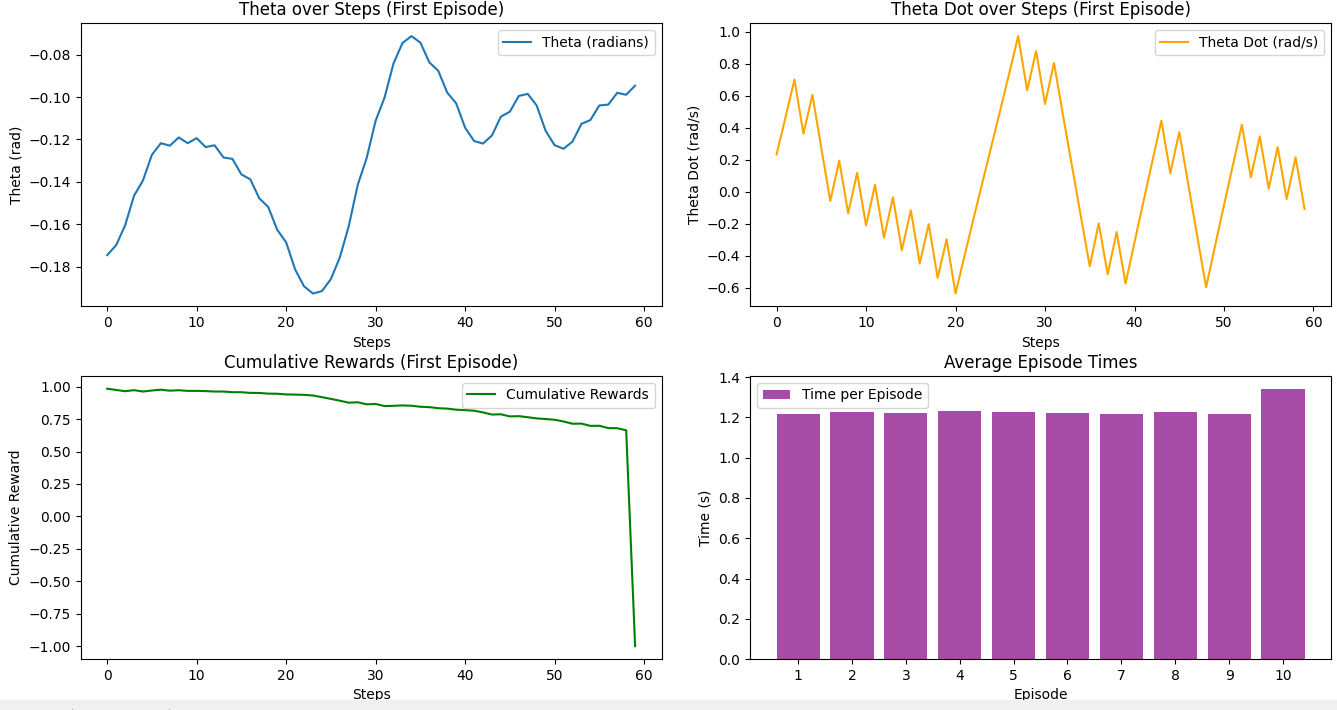
## **¿Cómo responde el agente a estados iniciales desfavorables, como un ángulo del péndulo cercano al límite de caída o a una posición extrema del carro?**

* **Entrenamiento de 5000 episodios con 20% de exploración y el péndulo inicia con un ángulo desfavorable cercano al ángulo límite de operación**

El ángulo posicionado está posicionado 11° de la posición vertical, el límite de operación es de 12° de la vertical. Inicialmente el ángulo desfavorable demora su aprendizaje. Observamos tendencia de mejora de política que significa que el agente logra aprender a estabilizarse aún inicializando en una posición desfavorable.



En el entrenamiento observamos un tiempo de estabilidad de 3 segundos. Aun así

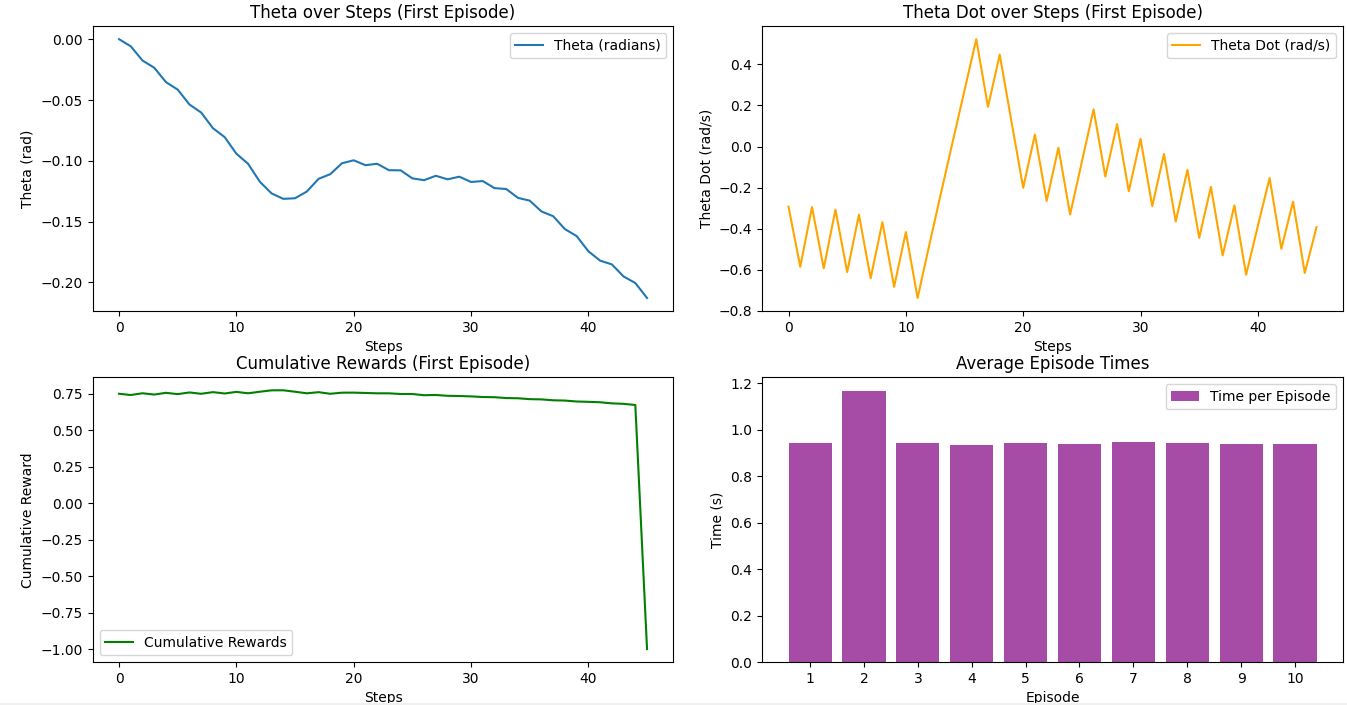


* **Entrenamiento de 5000 episodios con 20% de exploración y el carrito inicia en una posición extrema desfavorable cercana a la posición límite de operación**

El carrito está inicialmente en la posición -1m y la posición límite de operación está entre -1.2m y 1.2m. En la evolución de las recompensas en el entrenamiento obtenemos que no hay tendencia de mejora de la política además al inicio de la exploración no hay un progreso sostenido. Observamos que la posición desventajosa perjudica una rápida estabilidad. Además se observó que la fuerza de 1N aplicada es insuficiente para lograr alcanzar la posición vertical luego de alejarse de la posición desventajosa.



En el entrenamiento el carrito estabiliza el péndulo pero no considera permanecer en la posición central 0m ya que no se le ha recompensado suficientemente ese estado. El número de episodios es suficiente pero la política no está estructurada de manera que priorice la posición central.



* **Entrenamiento de 15000 episodios con 30% de exploración y una ligera recompensa por mantenerse en la posición inicial, una fuerza de 10N y una valoración mayor de las recompensas futuras.**

Ahora se recompensa bien el mantenerse en la posición central, además se valorará más las recompensas futuras ya que las acciones en el ambiente tienen consecuencias a largo plazo. Se aplica una fuerza de 10N para tener el torque necesario.

# **Conclusiones**

* El algoritmo Q-learning mostró su eficiencia para el aprendizaje de políticas de control en sistemas dinámicos como el péndulo invertido. A pesar de su simplicidad relativa en comparación con métodos más avanzados, el Q-learning logró estabilizar el sistema en diversas configuraciones, evidenciando su potencial para resolver problemas de control en entornos discretizados.
* La función de recompensa es un elemento determinante en la configuración del comportamiento del agente. Diseños que equilibran adecuadamente recompensas y penalizaciones contribuyen a políticas más estables. Recompensas que favorecen estados específicos, como mantener el carro en posición central o minimizar movimientos bruscos, impactaron positivamente en el aprendizaje y la estabilidad del sistema.
* La tasa de aprendizaje y el factor de descuento jugaron roles críticos en el desempeño del agente. Configuraciones con valores altos de mostraron mayor capacidad de adaptación y un enfoque en recompensas a largo plazo, mientras que tasas de aprendizaje moderadas evitaron oscilaciones significativas en el aprendizaje.
* Se observó que el agente puede aprender políticas efectivas incluso bajo condiciones iniciales desafiantes, como ángulos cercanos al límite de caída o posiciones extremas del carro. No obstante, estos escenarios requieren un mayor número de episodios de entrenamiento y ajustes específicos en las recompensas para garantizar un desempeño robusto.

# **Referencias**

Safeea, M. Neto, P. (2024). A Q-learning approach to the continuous control problem of robot inverted pendulum balancing. Intelligent Systems with Applications. 21:200313. <https://doi.org/10.1016/j.iswa.2023.200313>

Yildiran, U. (2024). Adaptive Control of an Inverted Pendulum by a Reinforcement Learning-based LQR Method. Yildiz Technical University, Control and Automation Engineering Department. <https://doi.org/10.48550/arXiv.2310.04436>

Kumar. S. (2020). Balancing a CartPole System with Reinforcement Learning -- A Tutorial. <https://doi.org/10.48550/arXiv.2006.04938>

# **Anexos**

* Código del proyecto implementado, ubicado en el repositorio del proyecto.