

**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

UNIVERSIDAD DEL PERÚ, DECANA DE AMÉRICA

**FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE**



---

## **PROYECTO FINAL - PÉNDULO INVERTIDO**

---

**ASIGNATURA** : Seguridad del Software

**DOCENTE** : ROSAS CUEVA, Yessica

**GRUPO** : 2

**PRESENTADO POR** : ANAYA SANCHEZ, Eros

ATUNCAR YATACO, Cristhian Paolo

LEON ROBLES, Illary Marcelo

LÓPEZ SALINAS, Marco Antonio

MARCHENA TEJADA, Taichi

**Lima, Perú**

**2024 - II**

# ÍNDICE

|                                     |           |
|-------------------------------------|-----------|
| <b>1. Objetivos del proyecto</b>    | <b>3</b>  |
| <b>2. Explicación de Q-learning</b> | <b>4</b>  |
| <b>3. Respuesta a las preguntas</b> | <b>6</b>  |
| 3.1. Pregunta A)                    | 6         |
| 3.2. Pregunta B)                    | 8         |
| 3.3. Pregunta C)                    | 10        |
| 3.4. Pregunta D)                    | 10        |
| 3.5. Pregunta E)                    | 11        |
| <b>4. Conclusiones</b>              | <b>13</b> |

# 1. Objetivos del proyecto

## 1.1. Objetivo general

Implementar y evaluar un sistema de control inteligente basado en aprendizaje por refuerzo clásico para estabilizar un péndulo invertido en un entorno simulado, maximizando su desempeño a través del diseño de políticas óptimas.

## 1.2. Objetivos específicos

- Diseñar un agente de aprendizaje por refuerzo utilizando algoritmos clásicos como Q-learning para estabilizar el péndulo invertido en el entorno CartPole-v1.
- Estudiar cómo la longitud del péndulo, la masa del carro y la fuerza máxima aplicable afectan la estabilidad y la capacidad del agente para aprender una política efectiva.
- Proponer una función de recompensa que fomente mantener el equilibrio del péndulo, penalizando movimientos excesivos del carro o la caída del péndulo, y evaluar su impacto en el aprendizaje del agente.
- Explorar estrategias para acelerar la convergencia del agente, como ajustes en la tasa de aprendizaje, el balance entre exploración y explotación, y el factor de descuento ( $\gamma$ ).
- Analizar el comportamiento del sistema ante condiciones iniciales desfavorables, como un ángulo cercano al límite de caída o una posición extrema del carro.

## 2. Explicación de Q-learning

El algoritmo **Q-learning** es una técnica de aprendizaje por refuerzo que permite a un agente aprender una política óptima para interactuar con un entorno. En el caso del péndulo invertido, el objetivo es aprender una política que mantenga el péndulo en posición vertical mientras se minimiza el movimiento del carro.

### 2.1. Representación del entorno

El entorno es el simulador **CartPole-v1** de Gym, que proporciona:

- Un estado continuo descrito por variables como la posición del carro, la velocidad, el ángulo del péndulo y su velocidad angular.
- Acciones discretas que representan las fuerzas aplicables al carro (moverlo a la izquierda o a la derecha).

### 2.2. Discretización del espacio de estados

Dado que Q-learning requiere un espacio de estados finito, se discretizan las observaciones continuas del entorno en un conjunto de estados discretos.

Esto simplifica la construcción de la tabla **Q**, que almacena los valores esperados de recompensa para cada combinación de estado y acción.

### 2.3. Función de recompensa

- El agente recibe una recompensa positiva por cada paso en el que mantiene el péndulo equilibrado.
- Penalizaciones severas se aplican cuando el péndulo cae o el carro se mueve fuera de los límites, incentivando al agente a evitar estas situaciones.

### 2.4. Actualización de la tabla Q

En cada iteración, el agente observa el estado actual **s**, selecciona una acción **a** y transita al siguiente estado **s'**. La tabla **Q** se actualiza usando la fórmula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_a Q(s', a) - Q(s, a)]$$

Donde:

- $\alpha$ : Tasa de aprendizaje.
- $\gamma$ : Factor de descuento que pondera las recompensas futuras.
- $r$ : Recompensa inmediata recibida.

## **2.5. Estrategia epsilon-greedy**

Durante el entrenamiento, el agente usa una política epsilon-greedy para balancear exploración y explotación:

Explora tomando acciones aleatorias con probabilidad  $\epsilon$ .

Explota el conocimiento adquirido seleccionando la acción con el mayor valor  $Q$  con probabilidad  $1-\epsilon$ .

## **2.6. Entrenamiento**

El agente aprende iterativamente mediante la interacción con el entorno. A medida que avanza el entrenamiento, la política mejora, reduciendo  $\epsilon$  para favorecer la explotación de la política aprendida.

## **2.7. Resultados esperados**

Tras el entrenamiento, el agente debería ser capaz de:

- Mantener el péndulo en posición vertical por períodos prolongados.
- Minimizar el movimiento del carro dentro de los límites del entorno.

### 3. Respuesta a las preguntas

#### 3.1. Pregunta A)

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

*En el siguiente fragmento de código, se especifica los valores de la longitud del péndulo, la fuerza máxima aplicable y la masa del carro:*

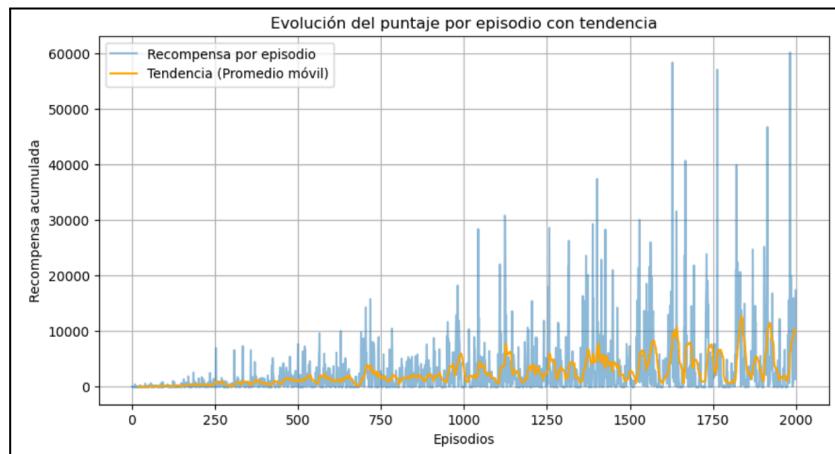
```
# Configuración inicial sin renderizado
def create_custom_env(length=0.5, mass_cart=1.0, force_mag=10.0, render_mode=None):
    env = gym.make('CartPole-v1', render_mode=render_mode)
    env.length = length
    env.env.masscart = mass_cart
    env.env.force_mag = force_mag
    return env
```

*Establecemos los siguientes como valores estándar para el algoritmo: 0.5 para la longitud, 1 para la masa del carro y 10 para la fuerza máxima. Veremos cómo influye la manipulación de estos factores en el entrenamiento del agente.*

##### **Manipulación de la fuerza máxima:**

- **Valor menor: Fuerza máxima = 7.5**

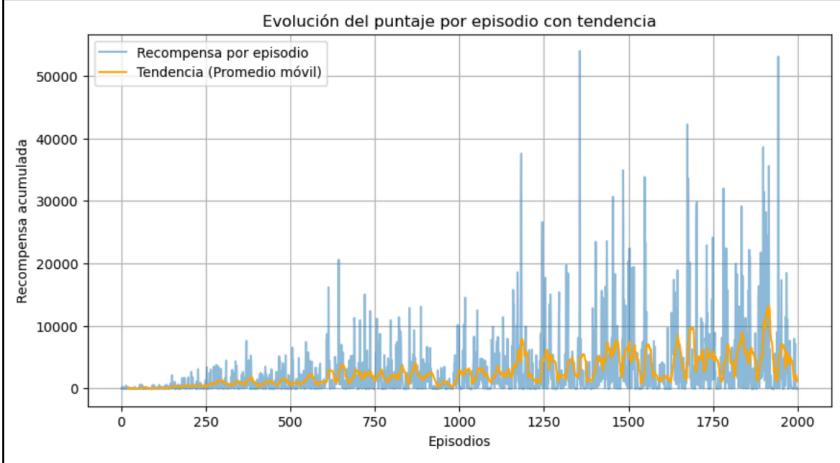
```
def create_custom_env(length=0.5, mass_cart=1.0, force_mag=7.5, render_mode=None):
    env = gym.make('CartPole-v1', render_mode=render_mode)
    env.length = length
    env.env.masscart = mass_cart
    env.env.force_mag = force_mag
    return env
```



*Vemos que con un valor menor las acciones posibles se vuelven limitadas, dificultando la recuperación de estados donde el carro está lejos del centro o el ángulo del péndulo es grande. Esto se puede apreciar en ambos gráficos, donde no hay un progreso estable en la consecución de episodios con altas recompensas.*

### - Valor mayor: Fuerza máxima= 16.0

```
# Configuración inicial sin renderizado
def create_custom_env(length=0.5, mass_cart=1.0, force_mag=16.0, render_mode=None):
    env = gym.make('CartPole-v1', render_mode=render_mode)
    env.env.length = length
    env.env.masscart = mass_cart
    env.env.force_mag = force_mag
    return env
```

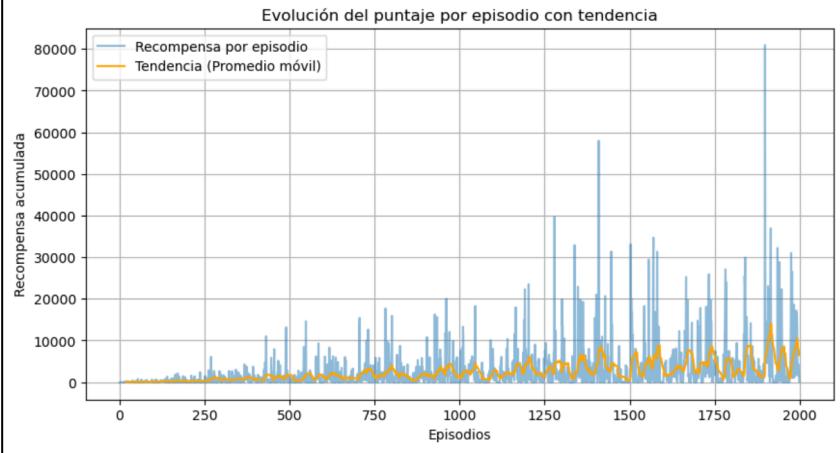


Una fuerza mayor permite al agente realizar correcciones más agresivas, ampliando su capacidad para estabilizar el sistema en situaciones extremas. Al permitir una mayor cantidad de fuerza, el sistema tiene más margen de acción para corregir su estado en cada uno de los episodios.

### Manipulación de la longitud del péndulo:

#### - Valor menor: 0.1

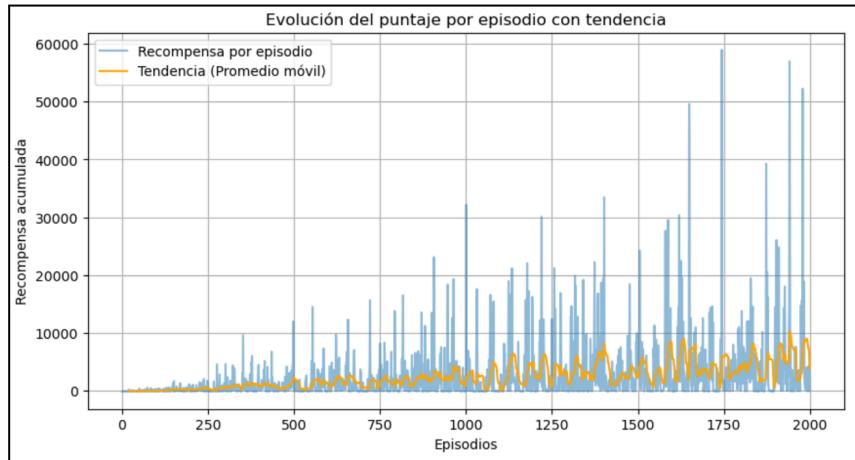
```
def create_custom_env(length=0.1, mass_cart=1.0, force_mag=10.0, render_mode=None):
    env = gym.make('CartPole-v1', render_mode=render_mode)
    env.env.length = length
    env.env.masscart = mass_cart
    env.env.force_mag = force_mag
    return env
```



*Un péndulo más corto incrementa la dificultad de estabilización, ya que tiene una mayor frecuencia de oscilación, lo que requiere que el agente tome decisiones rápidas y precisas.*

#### **- Valor mayor: Longitud = 2.0**

```
def create_custom_env(length=2.0, mass_cart=1.0, force_mag=10.0, render_mode=None):
    env = gym.make('CartPole-v1', render_mode=render_mode)
    env.env.length = length
    env.env.masscart = mass_cart
    env.env.force_mag = force_mag
    return env
```



*Un péndulo más largo reduce la complejidad del problema, ya que las acciones pueden ser más espaciadas y menos precisas, y existen menos penalizaciones por ángulos grandes. Es menos probable que el péndulo alcance rápidamente el límite de caída, debido a que oscila más lentamente, lo que da al agente más tiempo para corregir su posición.*

### **3.2. Pregunta B)**

**¿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?**

**Existen tres penalizaciones en este problema:**

- ❖ **Caída del péndulo: Penalización severa de -200.**
- ❖ **Ángulo del péndulo fuera del rango seguro ( $\pm 12^\circ$ ): Penalización moderada de -10.**

- ❖ Movimiento excesivo del carro ( $|x|$  grande): **Penalización proporcional al desplazamiento, definida como -  $\text{abs}(x) * 0.1$ .**

*Estas penalizaciones están diseñadas para guiar al agente hacia políticas que mantengan el péndulo equilibrado y el carro estable cerca del centro del riel, e influyen directamente en el valor de la recompensa (reward).*

*Penalizaciones en código:*

```
def custom_reward(obs, reward, done):
    x, x_dot, theta, theta_dot = obs
    if done:
        return -200
    if abs(theta) > math.radians(12):
        return -10
    return reward + (1 - abs(theta)) / math.radians(12) * 10 - abs(x) * 0.1
```

*Valores por defecto:*

**Caída de péndulo = -200**

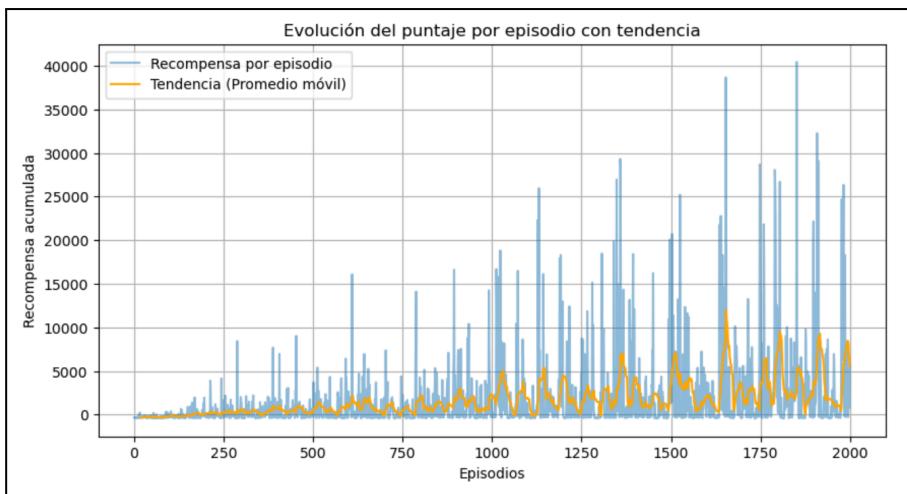
**Ángulo del péndulo fuera de rango = -10**

**Movimiento excesivo del carro = -  $\text{abs}(x) * 0.1$**

*Veamos cómo afecta la manipulación de estas penalizaciones en el resultado del algoritmo con el entorno estándar.*

**Caída de péndulo (penalización mayor) = - 500**

```
Episode 0: Total Reward: -344.2110286263686
Episode 25: Total Reward: -336.80522568063805
Episode 50: Total Reward: -400.121685721149
Episode 75: Total Reward: -155.68214321270574
Episode 100: Total Reward: -268.35735760459784
Episode 125: Total Reward: -424.9316363808834
Episode 150: Total Reward: 879.1313887339969
Episode 175: Total Reward: 536.496265413469
Episode 200: Total Reward: -268.7243956498718
Episode 225: Total Reward: -443.62977309168997
Episode 250: Total Reward: -325.9591957560315
Episode 275: Total Reward: -221.83312499545883
Episode 300: Total Reward: 1372.690177687756
Episode 325: Total Reward: -97.57015787151204
Episode 350: Total Reward: -253.98624134091273
Episode 375: Total Reward: 1090.759073397445
Episode 400: Total Reward: -362.008000997868396
Episode 425: Total Reward: 307.36785700415317
Episode 450: Total Reward: -224.49482897694793
Episode 475: Total Reward: 116.37979061584838
Episode 500: Total Reward: 2219.354169574396
Episode 525: Total Reward: -358.54525131064776
Episode 550: Total Reward: -386.719826109822
Episode 575: Total Reward: 1035.4164399199387
Episode 600: Total Reward: 1141.0160035397284
Episode 625: Total Reward: 2280.0461643191106
Episode 650: Total Reward: -270.72079745338885
Episode 675: Total Reward: 3559.0760841855367
```

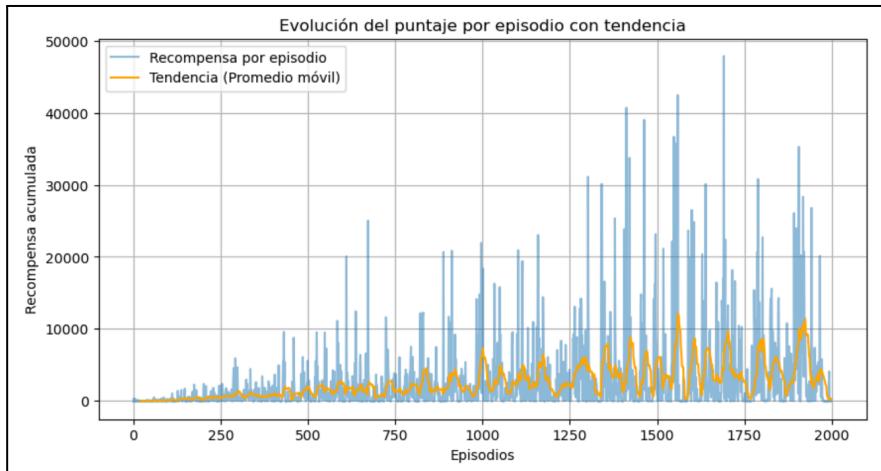


### **Ángulo del péndulo fuera de rango (penalización menor)= - 2**

```

Episode 0: Total Reward: -123.2973031833285
Episode 25: Total Reward: -85.99356318722994
Episode 50: Total Reward: -141.22733822067246
Episode 75: Total Reward: -42.27641269371577
Episode 100: Total Reward: 73.78700482836058
Episode 125: Total Reward: -57.387491236047936
Episode 150: Total Reward: 61.17068379208234
Episode 175: Total Reward: 744.4855868003635
Episode 200: Total Reward: 92.30823633126778
Episode 225: Total Reward: 52.53366510888054
Episode 250: Total Reward: 1054.3823419110215
Episode 275: Total Reward: 290.8351428808392
Episode 300: Total Reward: 1067.883014909396
Episode 325: Total Reward: 796.8972322671566
Episode 350: Total Reward: 984.08299829062
Episode 375: Total Reward: 156.6335089535758
Episode 400: Total Reward: 1361.867916116912
Episode 425: Total Reward: 38.81676255983811
Episode 450: Total Reward: -10.793738429202904
Episode 475: Total Reward: 610.1218222969693
Episode 500: Total Reward: 2667.6992631793814
Episode 525: Total Reward: 9500.425166537847
Episode 550: Total Reward: 216.5361364931511
Episode 575: Total Reward: 3134.87677242589
Episode 600: Total Reward: 941.3253704669985
Episode 625: Total Reward: 68.11379197566532
Episode 650: Total Reward: 6359.235309854189
Episode 675: Total Reward: 1364.9403764971858
Episode 700: Total Reward: -30.740281833035368
Episode 725: Total Reward: 5742.786645999377
...

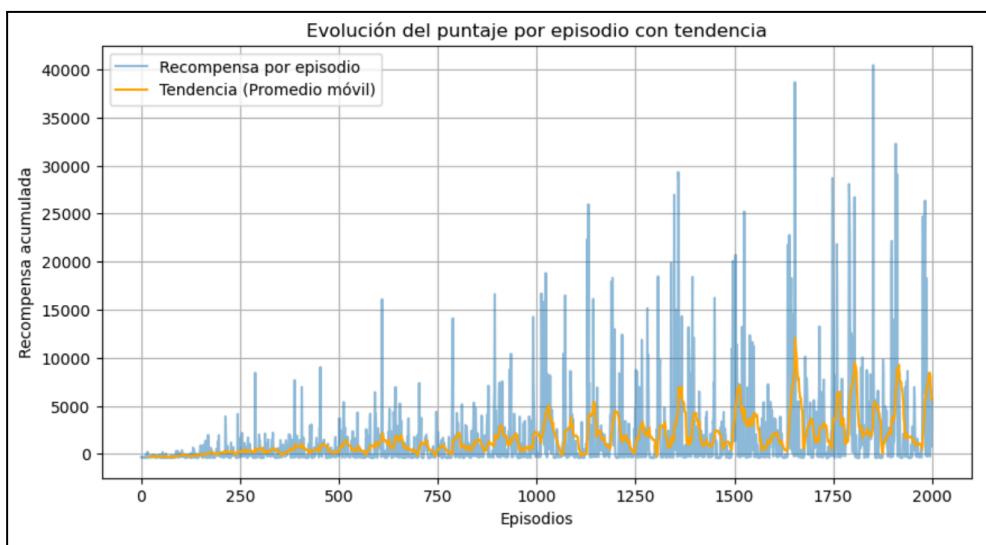
```



### **Movimiento excesivo del carro y Caída del Péndulo**

- 1. Movimiento excesivo (penalización mayor) = - abs(x) \* 1**
- 2. Caída del péndulo (penalización mayor) = - 500**

```
Episode 0: Total Reward: 106.69217118732018
Episode 25: Total Reward: -36.68851595124943
Episode 50: Total Reward: -32.046872702660124
Episode 75: Total Reward: 355.64853702649475
Episode 100: Total Reward: 376.23635084322757
Episode 125: Total Reward: -79.80249607597565
Episode 150: Total Reward: 385.32303858553223
Episode 175: Total Reward: -153.54292532840927
Episode 200: Total Reward: 774.3882462940728
Episode 225: Total Reward: 557.4088663626454
Episode 250: Total Reward: 2271.8663013610308
Episode 275: Total Reward: 639.2562582765364
Episode 300: Total Reward: 267.98572837612795
Episode 325: Total Reward: 443.6412570862034
Episode 350: Total Reward: 727.9044920603324
Episode 375: Total Reward: -74.42882188745492
Episode 400: Total Reward: -142.5857436547101
Episode 425: Total Reward: -79.76745789102094
Episode 450: Total Reward: 4.7137960432608565
Episode 475: Total Reward: -89.8634384379805
Episode 500: Total Reward: 95.95203571867467
Episode 525: Total Reward: 266.6344801496864
Episode 550: Total Reward: 1735.3034214609102
Episode 575: Total Reward: -91.2583574111696
Episode 600: Total Reward: 10524.889118996689
Episode 625: Total Reward: 378.33742044646874
Episode 650: Total Reward: 2778.3693053004417
Episode 675: Total Reward: 1483.538137329113
Episode 700: Total Reward: 3778.8154078951507
Episode 725: Total Reward: -109.34861851224314
```



*Efecto de penalizaciones:*

- *Penalizaciones más fuertes por movimientos excesivos del carro o ángulos grandes (como incrementar el factor de  $-abs(x)$  o el castigo por  $\theta > 12^\circ$ ): Enfocan al agente en minimizar estos errores, pero pueden conducir a un comportamiento conservador, sacrificando estabilidad general.*
- *Penalizaciones más suaves: Permiten más flexibilidad en los movimientos, pero podrían llevar al agente a encontrar soluciones menos óptimas.*

*En líneas generales, un valor muy alto o muy bajo de penalizaciones provocan que el sistema sea más propenso a desprender resultados irregulares y dificultan la tarea de alcanzar el equilibrio constante del péndulo.*

### 3.3. Pregunta C)

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

*En los siguientes funciones, se definen la tasa de aprendizaje, el factor de descuento y la política de exploración:*

```
def new_Q_value(reward: float, new_state: tuple, discount_factor=0.95) -> float:
    future_optimal_value = np.max(Q_table[new_state])
    return reward + discount_factor * future_optimal_value

def learning_rate(n: int, min_rate=0.01, max_rate=0.5) -> float:
    return max(min_rate, max_rate / (1 + 0.01 * n))

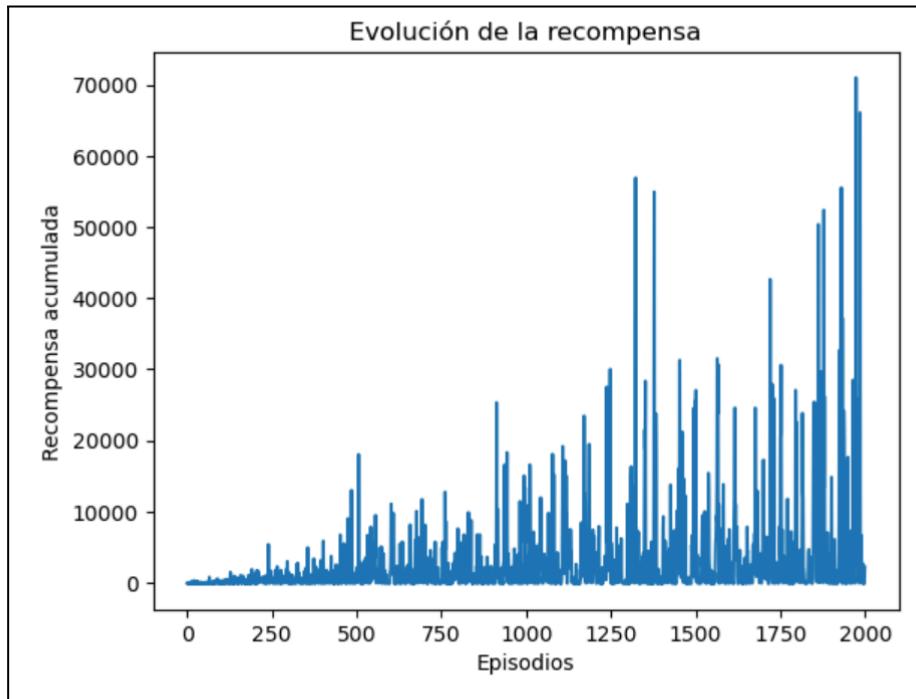
def exploration_rate(n: int, min_rate=0.1, max_rate=1.0) -> float:
    return max(min_rate, max_rate / (1 + 0.005 * n))
```

*Cambiaremos estos valores para analizar cómo afecta la variación al tiempo de convergencia.*

## Learning rate

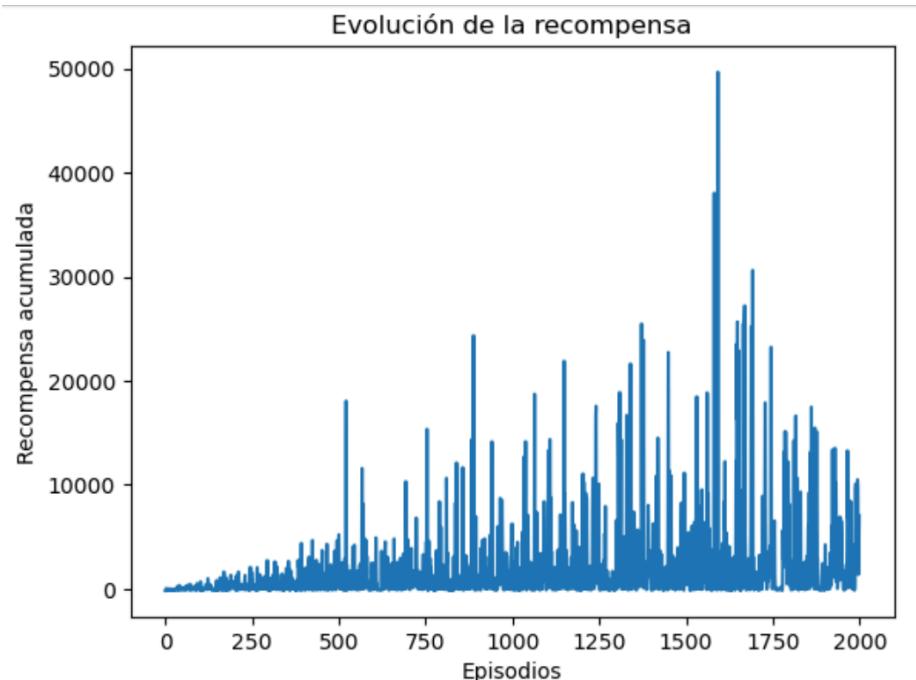
- Reducir min rate

```
def learning_rate(n: int, min_rate=0.004, max_rate=0.5) -> float:  
    return max(min_rate, max_rate / (1 + 0.01 * n))
```



- Aumentar max rate

```
def learning_rate(n: int, min_rate=0.01, max_rate=0.8) -> float:  
    return max(min_rate, max_rate / (1 + 0.01 * n))
```



*Al aumentar max\_rate...*

- Hace que los valores de la tabla Q se actualicen más rápidamente al inicio.
- Riesgo: Puede provocar inestabilidad en el aprendizaje si los valores se ajustan demasiado agresivamente.

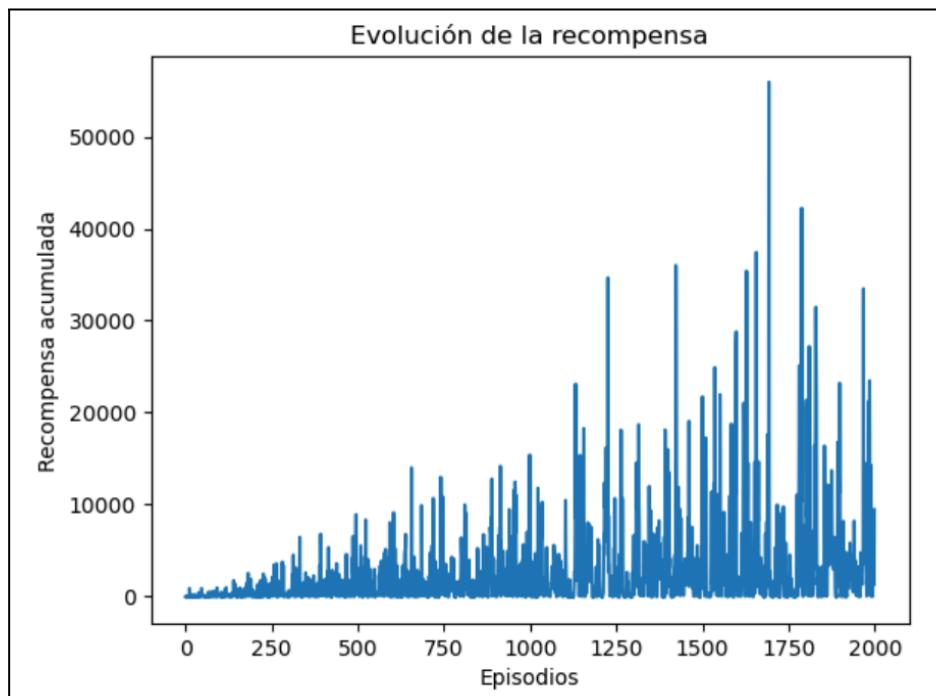
*Al disminuir min\_rate:*

- Reduce la tasa mínima, permitiendo ajustes más precisos en etapas tardías del aprendizaje.
- Riesgo: Un aprendizaje más lento en episodios avanzados.

### Discount factor

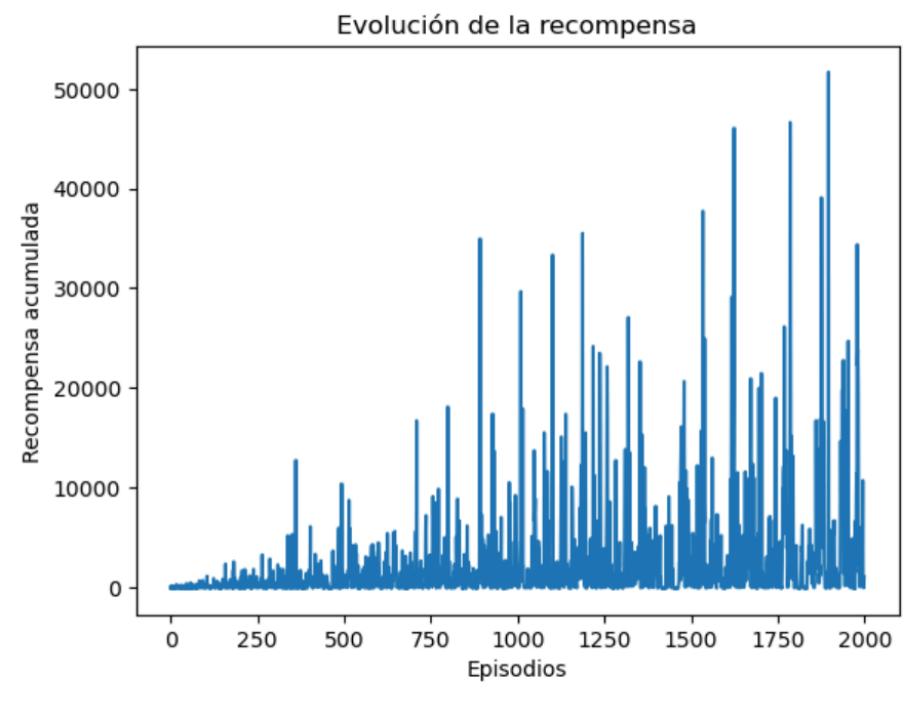
- Cercano a 0

```
def new_Q_value(reward: float, new_state: tuple, discount_factor=0.3) -> float:
```



- Cercano a 1

```
def new_Q_value(reward: float, new_state: tuple, discount_factor=0.99) -> float:
```



Al aumentar (cercano a 1)...

- El agente prioriza las recompensas futuras.
- Riesgo: El agente puede demorarse en aprender políticas óptimas inmediatas.

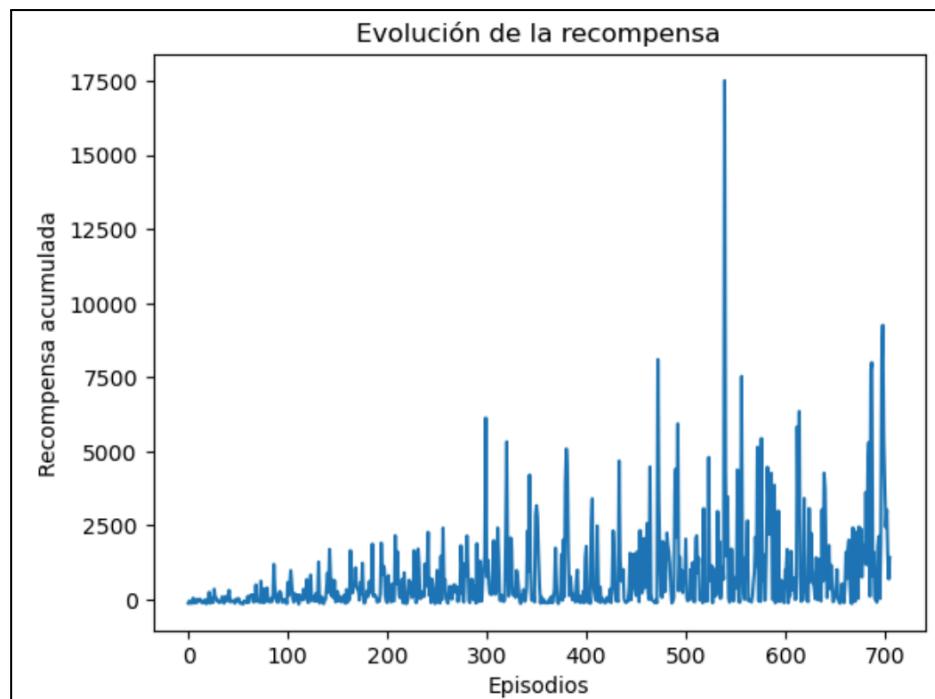
Al disminuir (cercano a 0)...

- El agente prioriza recompensas inmediatas.
- Riesgo: Puede llevar a políticas miope, ignorando recompensas de largo plazo

### Exploration rate

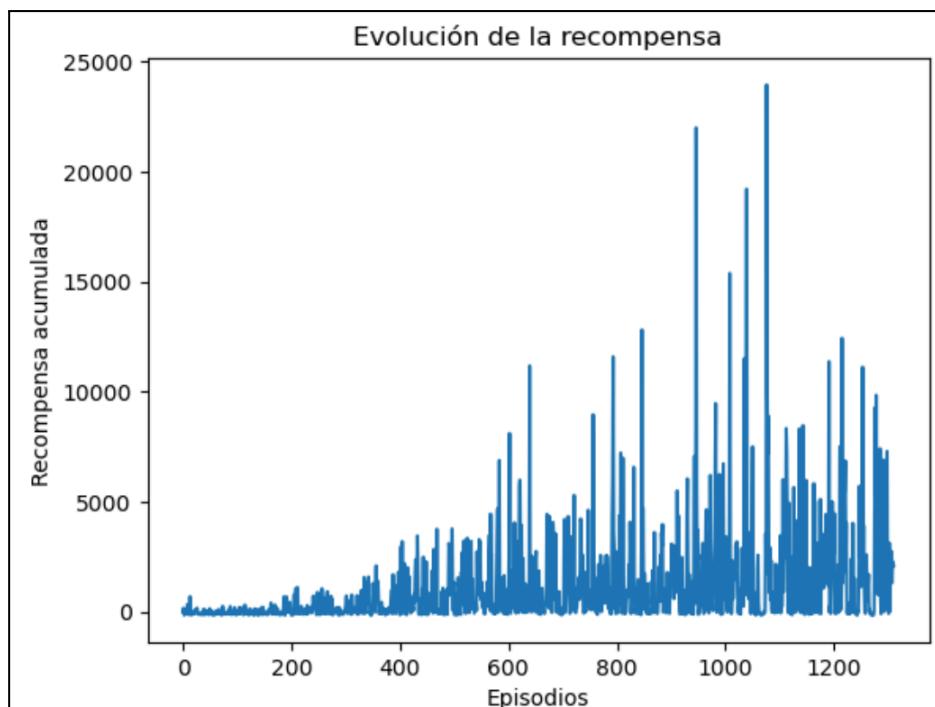
#### Cambio de min rate

```
def exploration_rate(n: int, min_rate=0.03, max_rate=1.0) -> float:  
    return max(min_rate, max_rate / (1 + 0.005 * n))
```



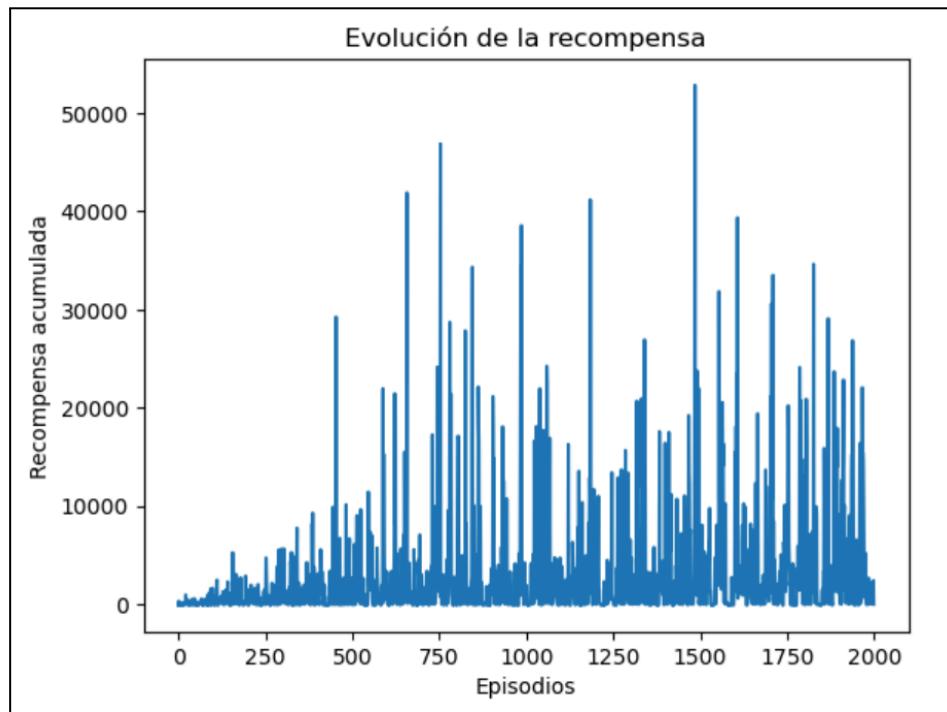
### Cambio de max rate

```
def exploration_rate(n: int, min_rate=0.1, max_rate=1.4) -> float:
    return max(min_rate, max_rate / (1 + 0.005 * n))
```



### Cambio de velocidad de decrecimiento

```
def exploration_rate(n: int, min_rate=0.1, max_rate=1.0) -> float:  
    return max(min_rate, max_rate / (1 + 0.012 * n))
```



*Un nivel alto de exploración favorece el descubrimiento de estrategias óptimas en entornos complejos, mientras que una rápida disminución acelera la convergencia, pero podría limitar al agente a políticas subóptimas.*

#### 3.4. Pregunta D)

**¿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?**

- *Si el ángulo inicial está cerca de  $\pm 12^\circ$  (el límite establecido), el agente intentará rápidamente aplicar fuerza en la dirección correcta para reducir el ángulo.*
- *Si el carro comienza cerca de los límites del riel, el agente podría tener dificultades para mantener el péndulo en equilibrio porque cualquier corrección adicional podría hacer que el carro se salga del riel.*

*De igual manera, la capacidad del agente depende en gran medida de cómo se ha explorado y aprendido en situaciones similares durante el entrenamiento.*

### 3.5. Pregunta E)

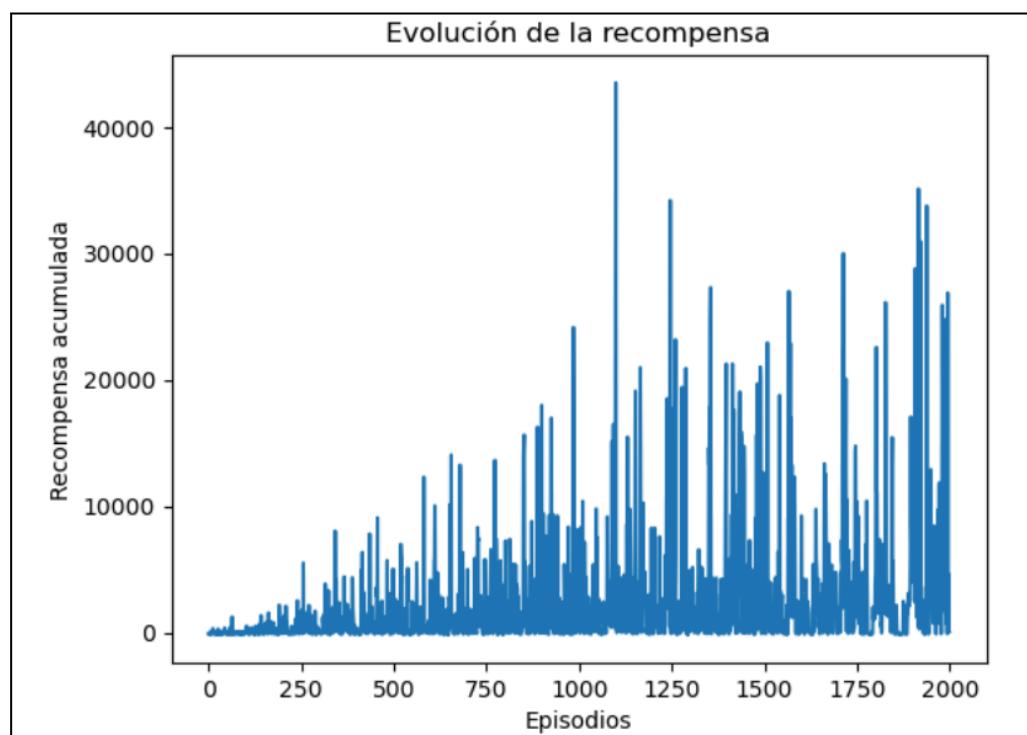
**Genera gráficos que representen la evolución de la recompensa acumulada por episodio durante el entrenamiento.**

*Con la librería “matplotlib” podemos generar gráficos para visualizar el progreso de nuestro algoritmo en cada episodio. Los siguientes son gráficos desprendidos del entorno con valores estándar (fuerza máxima, masa del carro, longitud del péndulo).*

**Gráfico 1:**

```
# Graficar evolución de la recompensa acumulada
plt.plot(rewards_per_episode)
plt.xlabel("Episodios")
plt.ylabel("Recompensa acumulada")
plt.title("Evolución de la recompensa")
plt.show()

# Cerrar el entorno al final
env.close()
```

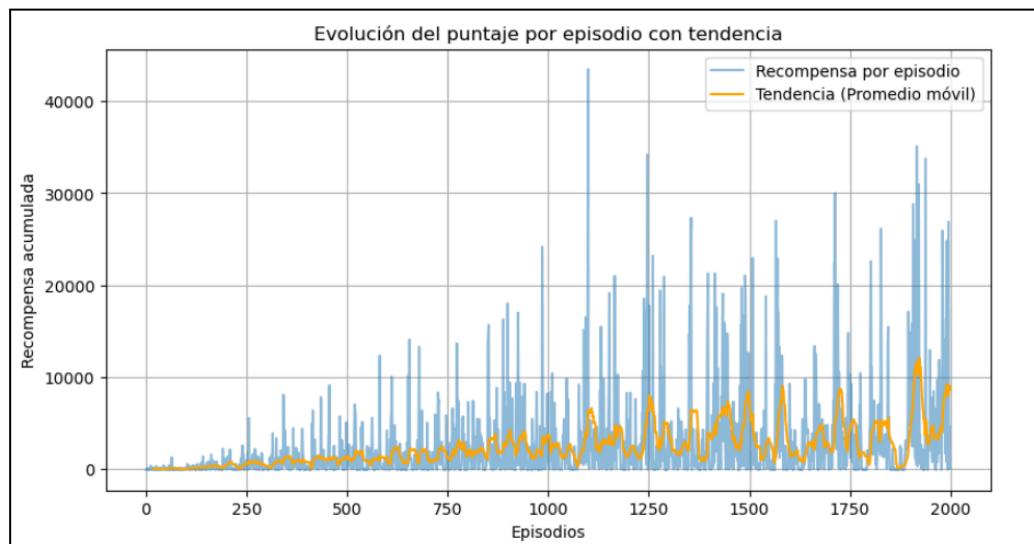


*En el eje Y, visualizamos la cantidad de recompensa acumulada en cada uno de los episodios (2000 en total), que están distribuidos en el eje X.*

**Gráfico 2:**

```
# Calcular el promedio móvil
window_size = 20 # Tamaño de la ventana
moving_avg = np.convolve(rewards_per_episode, np.ones(window_size)/window_size, mode='valid')

# Graficar recompensa acumulada y promedio móvil
plt.figure(figsize=(10, 5))
plt.plot(rewards_per_episode, label="Recompensa por episodio", alpha=0.5)
plt.plot(range(window_size - 1, len(rewards_per_episode)), moving_avg, label="Tendencia (Promedio móvil)", color="orange")
plt.xlabel("Episodios")
plt.ylabel("Recompensa acumulada")
plt.title("Evolución del puntaje por episodio con tendencia")
plt.legend()
plt.grid()
plt.show()
```



*En este caso, generamos un gráfico con más desarrollo, manteniendo los valores de la recompensa acumulada y los episodios en el eje Y y X respectivamente. El presente gráfico incluye la tendencia de las recompensas durante los episodios.*

*Se puede concluir, según los gráficos, que la evolución del algoritmo muestra una tendencia a producir picos más altos de recompensas acumuladas conforme avanza el entrenamiento. De igual manera, los valores altos de recompensa se mantienen con mayor regularidad progresivamente, lo que se ve reflejado en la línea de tendencia del gráfico 2.*

## **4. Conclusiones**

El proyecto demostró que el uso de Q-learning es efectivo para estabilizar un péndulo invertido, aunque depende significativamente de la configuración inicial de parámetros. Factores como la longitud del péndulo, la masa del carro y la fuerza máxima aplicable influyen en el desempeño, destacando que un péndulo más largo y mayor fuerza máxima facilitan la estabilización. El diseño de la función de recompensa resultó crucial, ya que penalizaciones bien equilibradas guiaron al agente hacia políticas óptimas, mientras que valores extremos dificultaron el aprendizaje. Además, se observó que ajustes en la tasa de aprendizaje, el balance entre exploración y explotación, y el factor de descuento impactaron directamente en el tiempo de convergencia, logrando mejoras progresivas reflejadas en el aumento de la recompensa acumulada y la estabilidad del sistema.