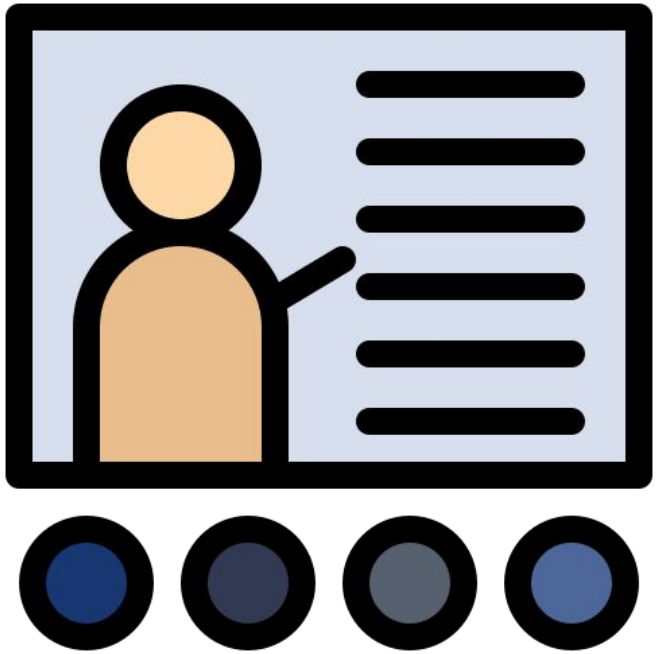
Modelado de Incertidumbre en Inteligencia Artificial

Solución del problema Cart-Pole evaluando algoritmos de Aprendizaje por Refuerzo

Ponentes:

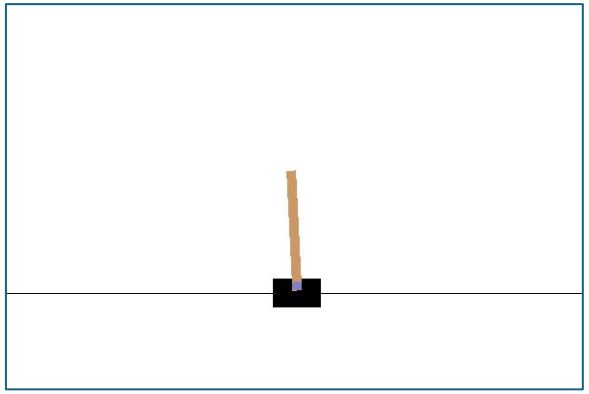
- **Juan Herencia**
- **Jeyson Lino**
- **José Zúñiga**

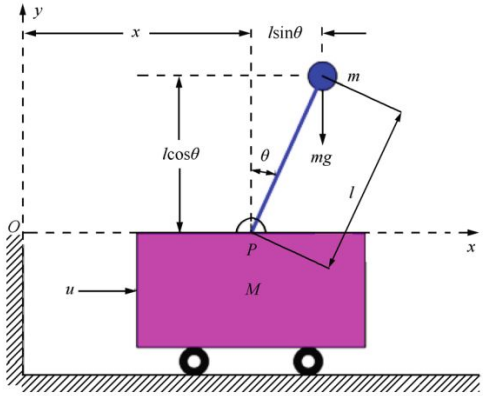
1. Introducción



Introducción

Descripción del problema





Cart-Pole es un péndulo con un centro de gravedad por encima de su punto de pivote. Es inestable, pero se puede controlar moviendo el punto de pivote por debajo del centro de masa.

El problema es la inestabilidad del poste cuando el carrito está en movimiento horizontal. La solución es mover el carrito de manera que el poste se mantenga en posición vertical.

Introducción

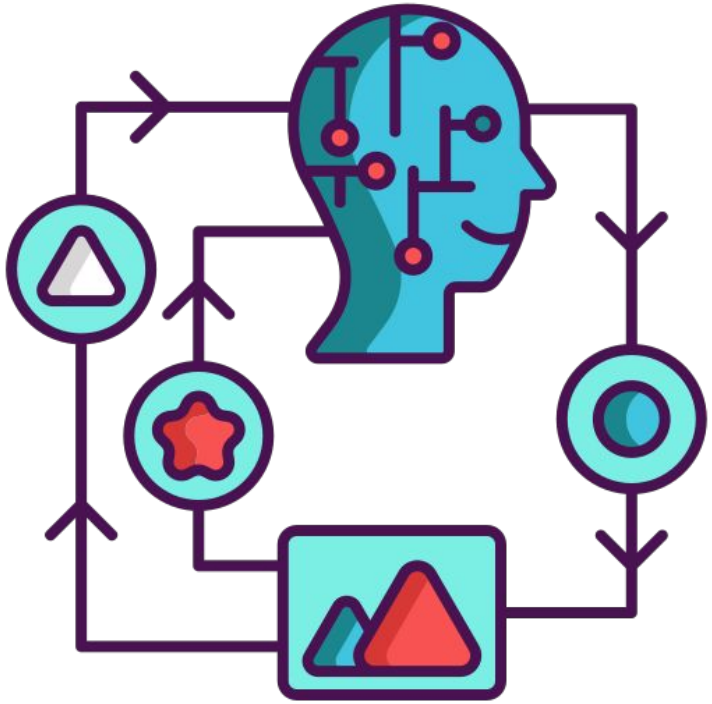
Objetivos del proyecto

Encontrar la solución al problema del Cart-Pole usando algoritmos de aprendizaje por refuerzo que puedan controlar sistemas dinámicos.

Criterio de éxito

El problema se considera resuelto si el poste se mantiene en equilibrio por una cierta cantidad de pasos de tiempo.

2. Adaptación del problema



Adaptación del problema

El objetivo es mover el carrito de manera que el poste se mantenga en posición vertical (o lo más cercano a la vertical posible) sin que se caiga.

Agente: sistema que permite el ingreso de valores

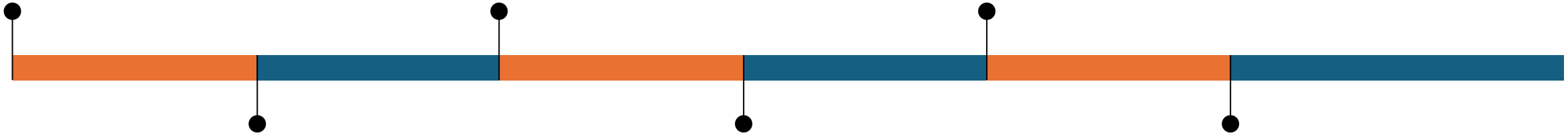
Entorno: carrito que se puede mover horizontalemente

Acción (a)

Mover el carrito hacia la izquierda o hacia la derecha, aplicando fuerzas

Política (Pi)

Estrategia que define qué acción tomar en cada estado



Estado (s)

- Posición del carrito
- Velocidad del carrito
- Ángulo del poste
- Velocidad angular del poste

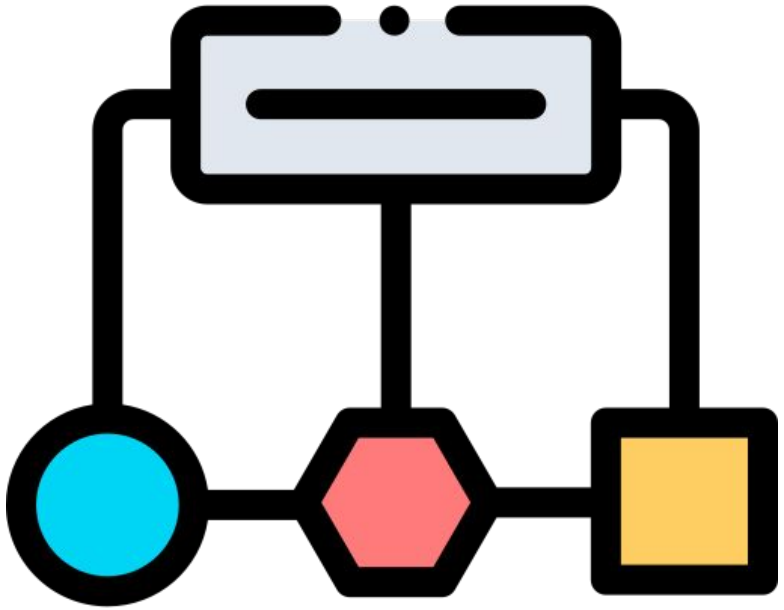
Recompensa (r)

Positiva si el poste está equilibrado y negativa si éste se cae

Valor (V): recompensa esperada desde un estado, siguiendo una política

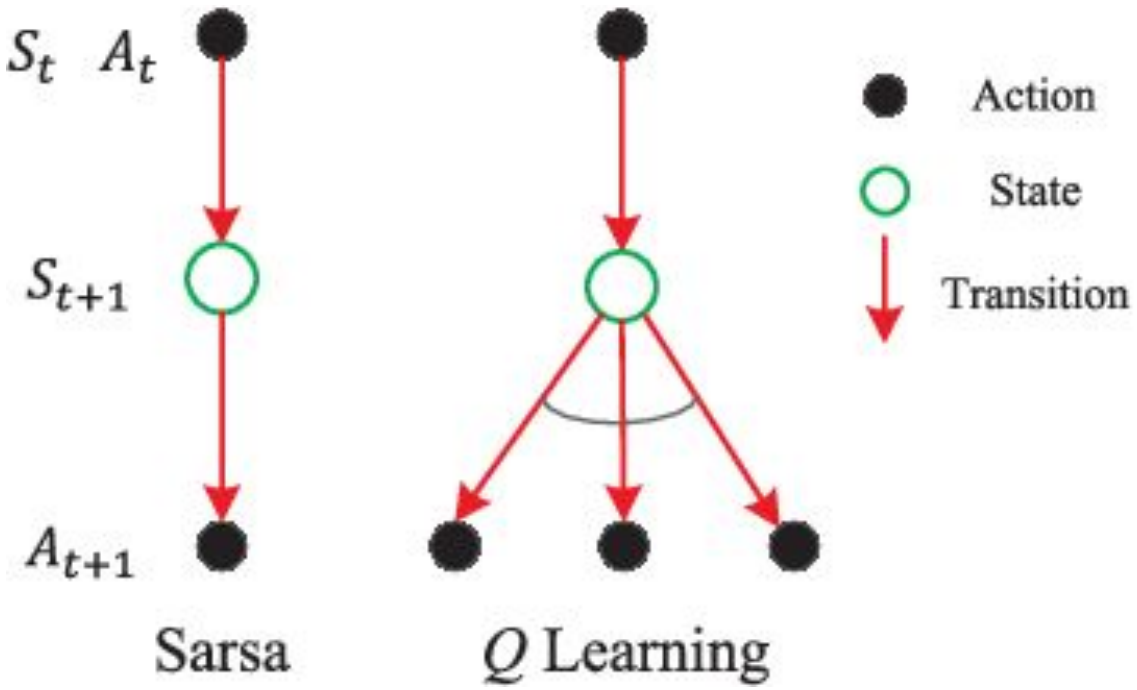
Función Q (Q): recompensa esperada de tomar una acción en un estado dado, luego de seguir una política

3. Metodología



Metodología

Selección de algoritmos



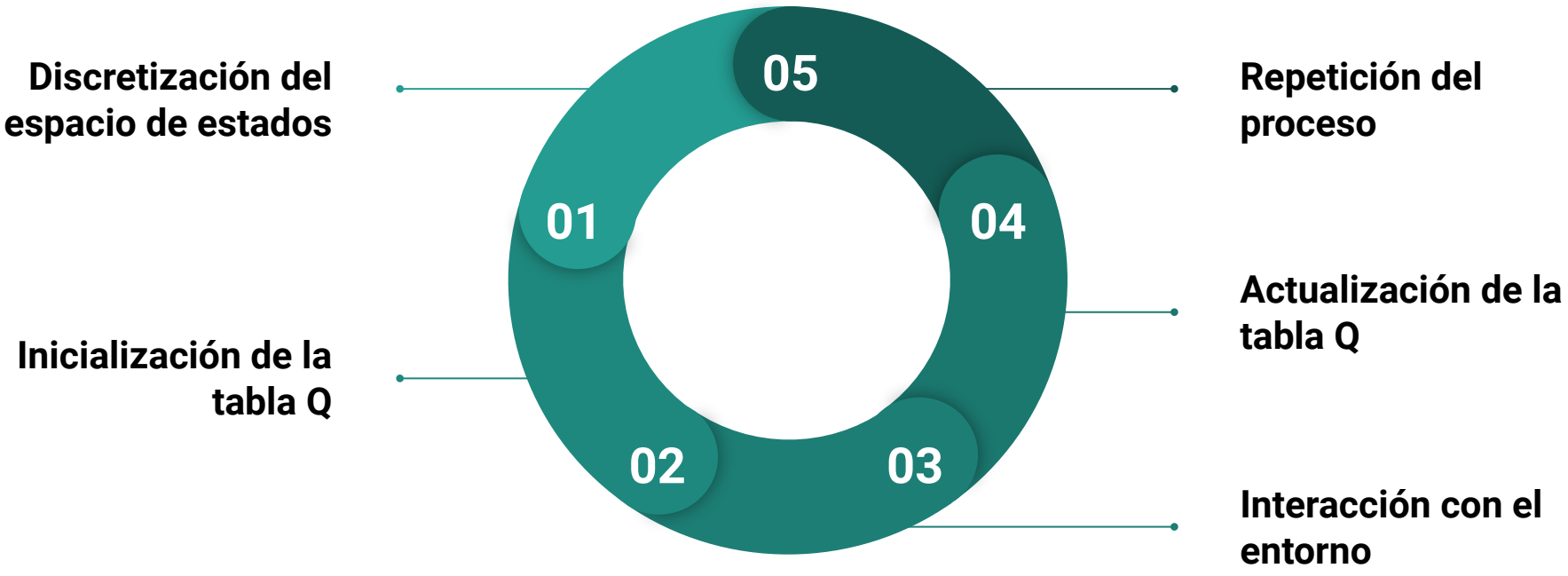
Con estos algoritmos se gestionarán las variables de estado:

- Posición del carrito
- Velocidad del carrito
- Ángulo del poste
- Velocidad angular del poste

Metodología

Algoritmo Q-Learning

$$Q(s_t, at_) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$$

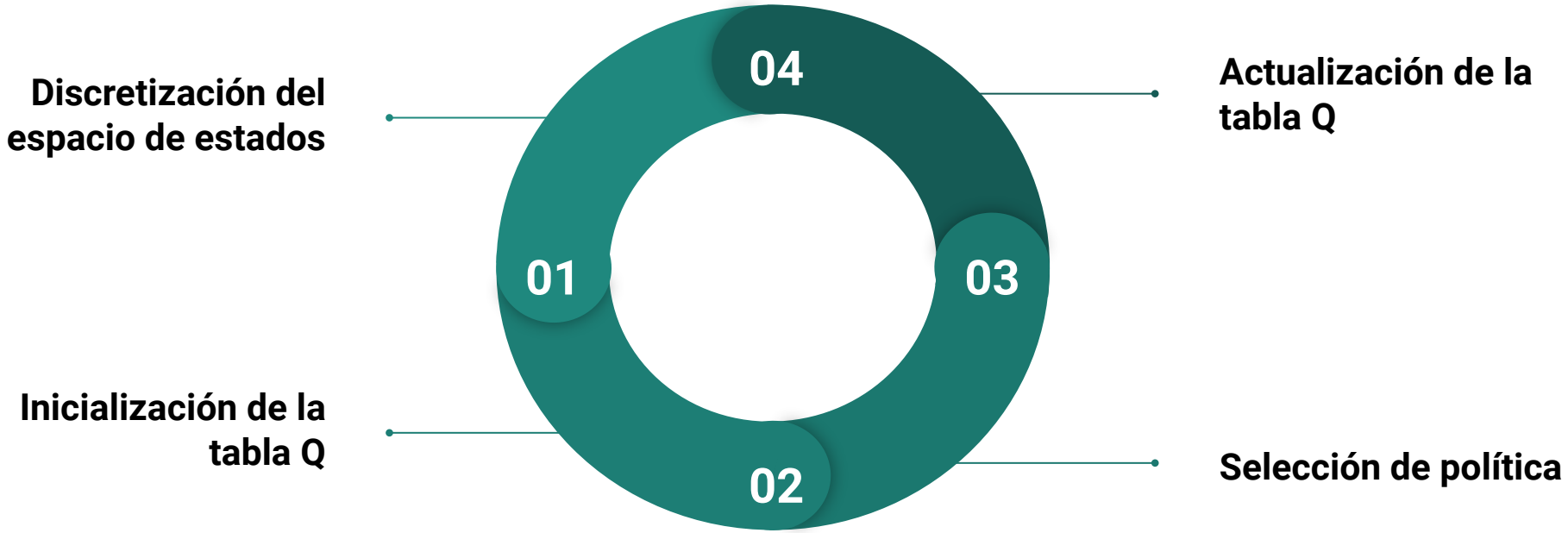


Pasos para adaptar el algoritmo Q-Learning

Metodología

Algoritmo Sarsa

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$



Pasos para adaptar el algoritmo Sarsa

4. Herramientas utilizadas

Gymnasium

Herramientas utilizadas

CartPole - v0

- Versión simple, adecuada para primeros experimentos en aprendizaje por refuerzo.
- Límite de episodios corto (200 pasos).
- El episodio termina si el poste se inclina más de 15 grados desde la vertical.
- El episodio termina si el carro se desplaza más de 2.4 unidades desde el centro.
- El episodio termina si el poste se mantiene en posición durante 200 pasos.
- La recompensa es +1 por cada paso de tiempo en el que el poste se mantiene en posición.

Herramientas utilizadas

CartPole - v1

- Límites más estrictos en ángulo del poste.
- Evaluación más exhaustiva del rendimiento del agente (500 pasos).
- Ideal para aplicaciones que han superado a CartPole - v0.
- El episodio termina si el poste se inclina más de 12 grados desde la vertical (más estricto que en v0).
- El episodio termina si el carro se desplaza más de 2.4 unidades desde el centro (igual que en v0).
- El episodio termina si el poste se mantiene en posición durante 500 pasos (más largo que en v0).
- La recompensa es +1 por cada paso de tiempo en el que el poste se mantiene en posición.

5. Implementación

Implementación

Q - Learning

```python
# Q-Learning
for episode in range(episodes):
    state = discretize_state(env.reset()[0])
    total_reward = 0
    consecutive_steps = 0

    for step in range(max_steps):
        if random.uniform(0, 1) < epsilon:
            action = env.action_space.sample()
        else:
            action = np.argmax(q_table[state])

        next_state, reward, done, _, _ = env.step(action)
        next_state = discretize_state(next_state)

        # Actualización de Q-Table
        q_table[state][action] = q_table[state][action] + learning_rate * (
            reward + discount_rate * np.max(q_table[next_state]) - q_table[state][action])

        state = next_state
        total_reward += reward

        # Verificar si el poste se ha mantenido en posición por más de 500 pasos consecutivos
        if abs(state[2]) < np.pi / 20:
            consecutive_steps += 1
        else:
            consecutive_steps = 0

        if consecutive_steps >= 500:
            done = True

        if done:
            break

    print(f'Episodio {episode}, Estado final: {state}, Recompensa total: {total_reward}')
```

```python
    # Reducción del valor de epsilon
    if epsilon > epsilon_min:
        epsilon *= epsilon_decay

    rewards_per_episode.append(total_reward)

    if episode % 100 == 0:
        print(f'Episodio: {episode}, Recompensa total: {total_reward}, Epsilon: {epsilon}')
```

Implementación

```python
# SARSA
for episode in range(episodes):
    state = discretize_state(env.reset()[0])
    total_reward = 0
    consecutive_steps = 0

    # Selección de la acción inicial
    if random.uniform(0, 1) < epsilon:
        action = env.action_space.sample()
    else:
        action = np.argmax(q_table[state])

    for step in range(max_steps):
        next_state, reward, done, _, _ = env.step(action)
        next_state = discretize_state(next_state)

        # Selección de la siguiente acción
        if random.uniform(0, 1) < epsilon:
            next_action = env.action_space.sample()
        else:
            next_action = np.argmax(q_table[next_state])

        # Actualización de Q-Table
        q_table[state][action] = q_table[state][action] + learning_rate * (
            reward + discount_rate * q_table[next_state][next_action] - q_table[state][action])

        state = next_state
        action = next_action
        total_reward += reward

        # Verificar si el poste se ha mantenido en posición por más de 500 pasos consecutivos
        if abs(state[2]) < np.pi / 20:
            consecutive_steps += 1
        else:
            consecutive_steps = 0

        if consecutive_steps >= 500:
            done = True

        if done:
            break

    print(f'Episodio {episode}, Estado final: {state}, Recompensa total: {total_reward}')
```

SARSA

```python
    # Reducción del valor de epsilon
    if epsilon > epsilon_min:
        epsilon *= epsilon_decay

    rewards_per_episode.append(total_reward)

    if episode % 100 == 0:
        print(f'Episodio: {episode}, Recompensa total: {total_reward}, Epsilon: {epsilon}')
```

6. Experimentación y resultados

Experimentación

1. Ecuaciones físico matemáticas

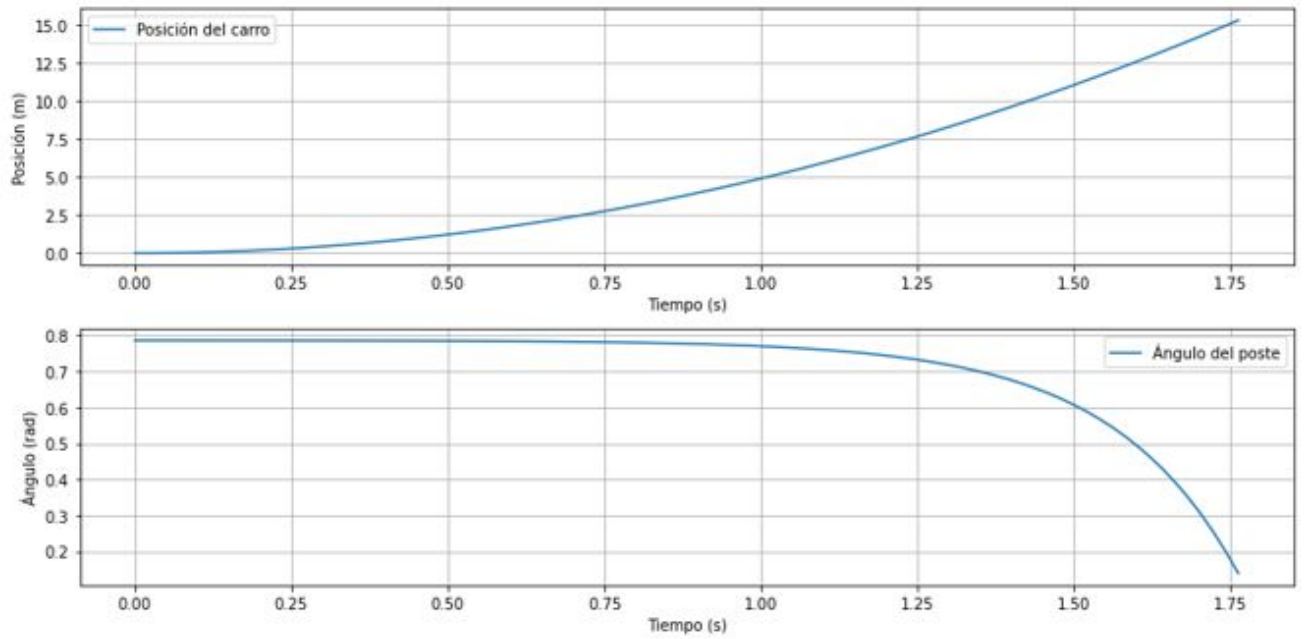
- En este caso solo se puede realizar un episodio por cada estado inicial que se indique.
- Hay convergencia con una fuerza de inicial de 12.76N

```
iteración 0-[ 1.96711136e-03  1.96514426e-01  7.85396909e-01 -1.25375740e-04]
iteración 50-[ 5.11764872 10.02885172  0.76769028 -0.08597444]
iteración 100-[20.0268162  18.22529945 -0.98485363 -5.38222423]
iteración 150-[37.81885633 18.27286444 -7.1339723  -5.13884912]
iteración 200-[ 55.60990179  18.30742556 -13.28957108  -4.91941127]
iteración 250-[ 73.40025888  18.3314357  -19.45096608  -4.72522064]
iteración 300-[ 91.19016649  18.34726103 -25.61742202  -4.55828056]
iteración 350-[108.97980082  18.35706188 -31.78814162  -4.42094956]
iteración 400-[126.76928369  18.3626926  -37.96225849  -4.3155666 ]
iteración 450-[144.55869325  18.36560688 -44.13884866  -4.24416394]
iteración 500-[162.34807128  18.36680524 -50.31693878  -4.20822087]
iteración 550-[180.13744083  18.36679655 -56.49552446  -4.20850717]
iteración 600-[197.92681825  18.36557392 -62.6735943   -4.24503099]
iteración 650-[215.71622722  18.36262409 -68.85014509  -4.31698987]
iteración 700-[233.50571153  18.35693908 -75.02420023  -4.42290719]
iteración 750-[251.29534866  18.34705171 -81.19483434  -4.56076777]
iteración 800-[269.08526152  18.33110516 -87.3611854   -4.72821279]
iteración 850-[286.87562742  18.30693099 -93.52245375  -4.92289332]
iteración 900-[304.66668685  18.27215526 -99.67790255  -5.14282464]
iteración 950-[ 322.45874662   18.22432243 -105.82685006   -5.38670456]
Resultado esperado en Iteración 88
Posicion del carro = 15.3 metros
Angulo de poste = 0.139 radianes
Tiempo de 1000 episodios      = 161.2548828125 segundos
```

Experimentación

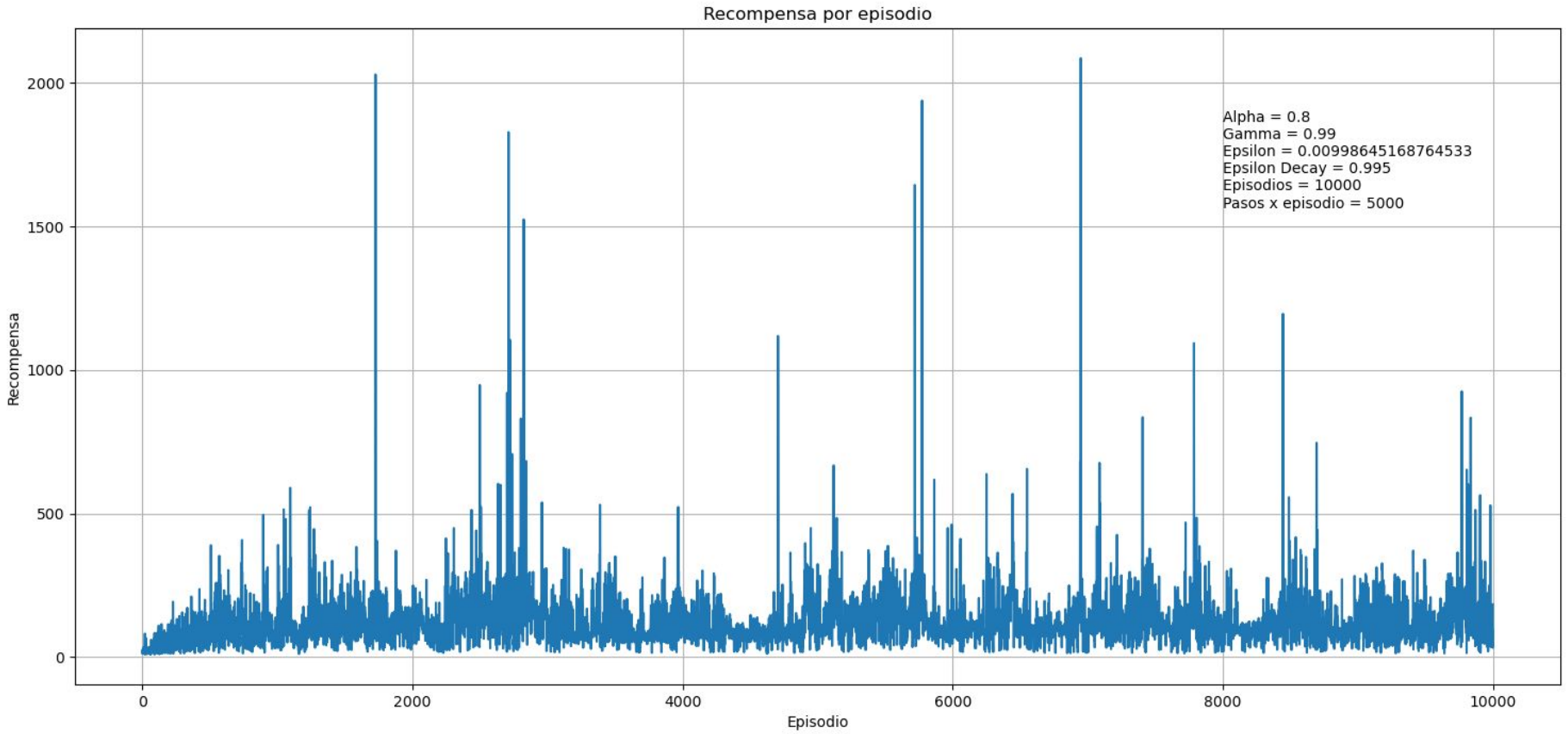
1. Ecuaciones físico matemáticas

- En este caso solo se puede realizar un episodio por cada estado inicial que se indique.
- Hay convergencia con una fuerza de inicial de 12.76N



Experimentación

2. Algoritmo Q-Learning



Experimentación

2. Algoritmo Q-Learning

```
Resultados para Q-Learning
Alpha = 0.8
Gamma = 0.99
Epsilon = 1.0
Epsilon Decay = 0.995
Episodios = 10000
Pasos x episodio = 5000
Episodio 1046, Estado final: (9, 8, 14, 11), Recompensa total: 515.0
Episodio 1094, Estado final: (8, 5, 14, 14), Recompensa total: 590.0
Episodio 1234, Estado final: (8, 8, 14, 10), Recompensa total: 511.0
Episodio 1242, Estado final: (10, 6, 15, 14), Recompensa total: 522.0
Episodio 1726, Estado final: (9, 9, 4, 6), Recompensa total: 2030.0
Episodio 2437, Estado final: (9, 10, 4, 6), Recompensa total: 513.0
Episodio 2498, Estado final: (7, 1, 14, 16), Recompensa total: 948.0
Episodio 2506, Estado final: (9, 8, 4, 8), Recompensa total: 523.0
Episodio 2634, Estado final: (7, 3, 15, 16), Recompensa total: 604.0
Episodio 2652, Estado final: (8, 2, 14, 16), Recompensa total: 600.0
Episodio 2700, Estado final: (8, 4, 15, 15), Recompensa total: 921.0
Episodio 2711, Estado final: (9, 8, 4, 9), Recompensa total: 1829.0
Episodio 2713, Estado final: (7, 2, 14, 15), Recompensa total: 692.0
Episodio 2722, Estado final: (9, 4, 14, 15), Recompensa total: 1105.0
Episodio 2736, Estado final: (7, 2, 14, 15), Recompensa total: 708.0
Episodio 2802, Estado final: (9, 9, 14, 11), Recompensa total: 831.0
Episodio 2823, Estado final: (7, 1, 14, 16), Recompensa total: 1525.0
Episodio 2842, Estado final: (8, 2, 14, 16), Recompensa total: 683.0
Episodio 2958, Estado final: (7, 1, 14, 16), Recompensa total: 539.0
Episodio 3388, Estado final: (8, 5, 14, 15), Recompensa total: 531.0
```

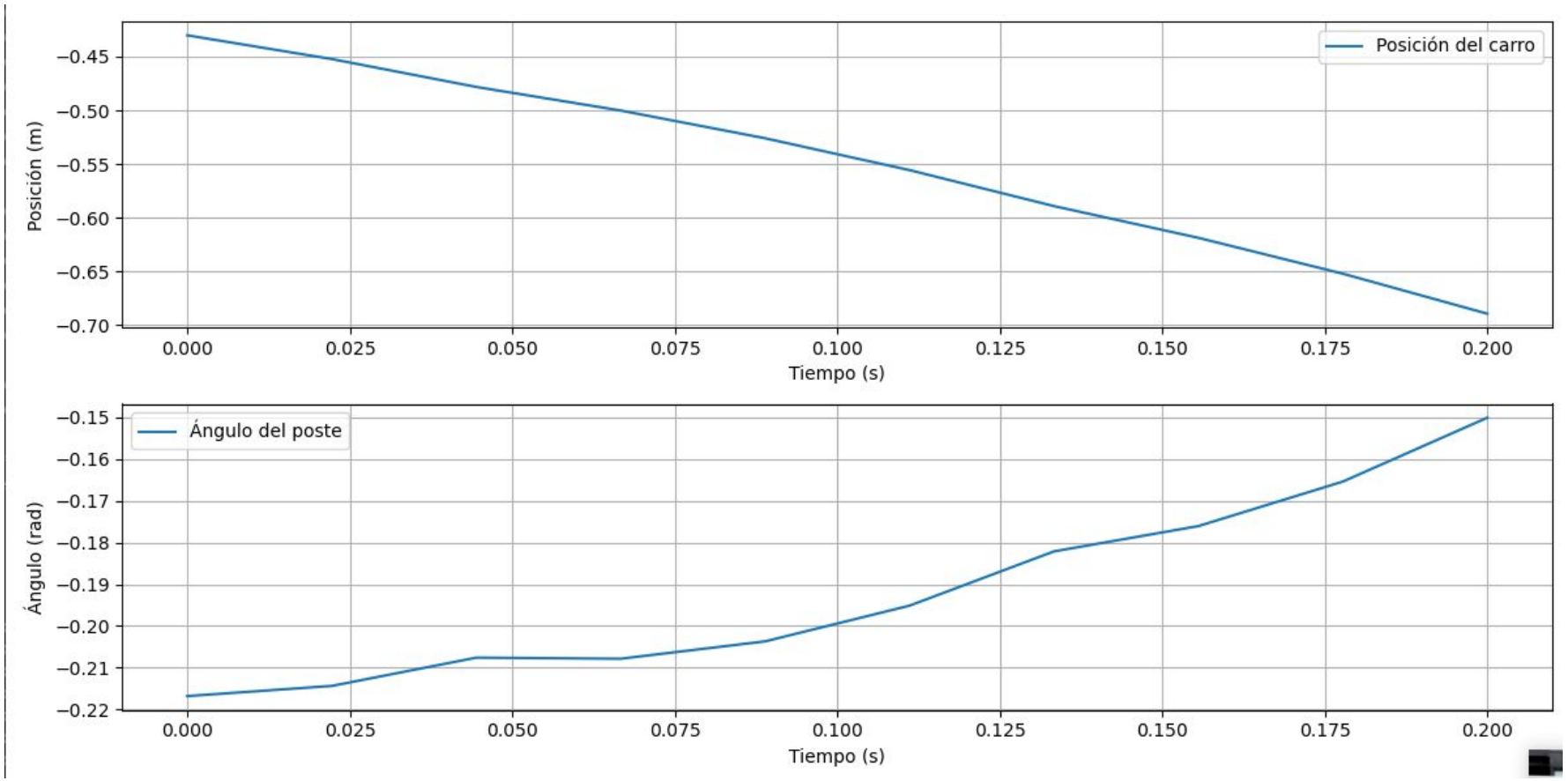
Experimentación

2. Algoritmo Q-Learning

```
Episodio 3967, Estado final: (8, 5, 14, 13), Recompensa total: 523.0
Episodio 4706, Estado final: (6, 1, 15, 16), Recompensa total: 1119.0
Episodio 4707, Estado final: (6, 0, 15, 17), Recompensa total: 697.0
Episodio 5118, Estado final: (9, 7, 4, 9), Recompensa total: 668.0
Episodio 5719, Estado final: (6, 1, 15, 16), Recompensa total: 1645.0
Episodio 5773, Estado final: (6, 0, 14, 17), Recompensa total: 1939.0
Episodio 5863, Estado final: (9, 7, 4, 8), Recompensa total: 618.0
Episodio 6250, Estado final: (6, 0, 14, 16), Recompensa total: 638.0
Episodio 6443, Estado final: (6, 1, 14, 16), Recompensa total: 569.0
Episodio 6550, Estado final: (8, 8, 14, 11), Recompensa total: 656.0
Episodio 6943, Estado final: (5, 0, 14, 17), Recompensa total: 678.0
Episodio 6948, Estado final: (6, 1, 14, 16), Recompensa total: 2087.0
Episodio 7087, Estado final: (10, 4, 14, 16), Recompensa total: 677.0
Episodio 7089, Estado final: (11, 6, 14, 14), Recompensa total: 537.0
Episodio 7406, Estado final: (7, 1, 14, 16), Recompensa total: 836.0
Episodio 7786, Estado final: (10, 6, 14, 14), Recompensa total: 1094.0
Episodio 8445, Estado final: (10, 4, 14, 15), Recompensa total: 1196.0
Episodio 8489, Estado final: (9, 2, 14, 16), Recompensa total: 557.0
Episodio 8694, Estado final: (10, 6, 14, 14), Recompensa total: 747.0
Episodio 9769, Estado final: (8, 7, 4, 8), Recompensa total: 926.0
Episodio 9806, Estado final: (9, 7, 4, 7), Recompensa total: 653.0
Episodio 9815, Estado final: (10, 3, 14, 16), Recompensa total: 603.0
Episodio 9835, Estado final: (9, 7, 4, 7), Recompensa total: 834.0
Episodio 9870, Estado final: (8, 6, 4, 9), Recompensa total: 513.0
Episodio 9905, Estado final: (10, 3, 14, 16), Recompensa total: 564.0
Episodio 9981, Estado final: (8, 9, 14, 11), Recompensa total: 529.0
Tiempo usado por Q-Learning con 10000 episodios = 57.74 segundos
```

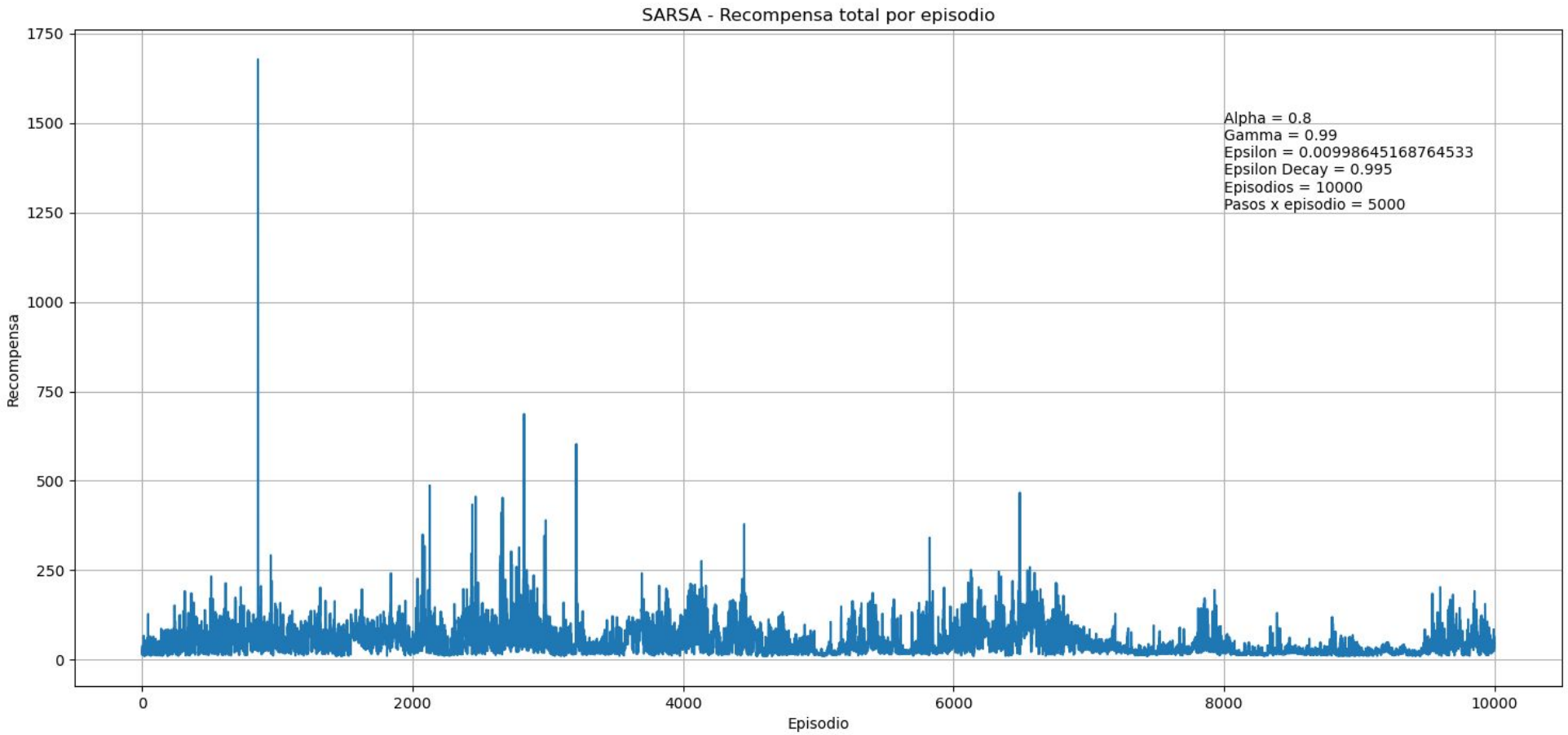
Experimentación

2. Algoritmo Q-Learning - Simulación



Experimentación

3. Algoritmo Sarsa



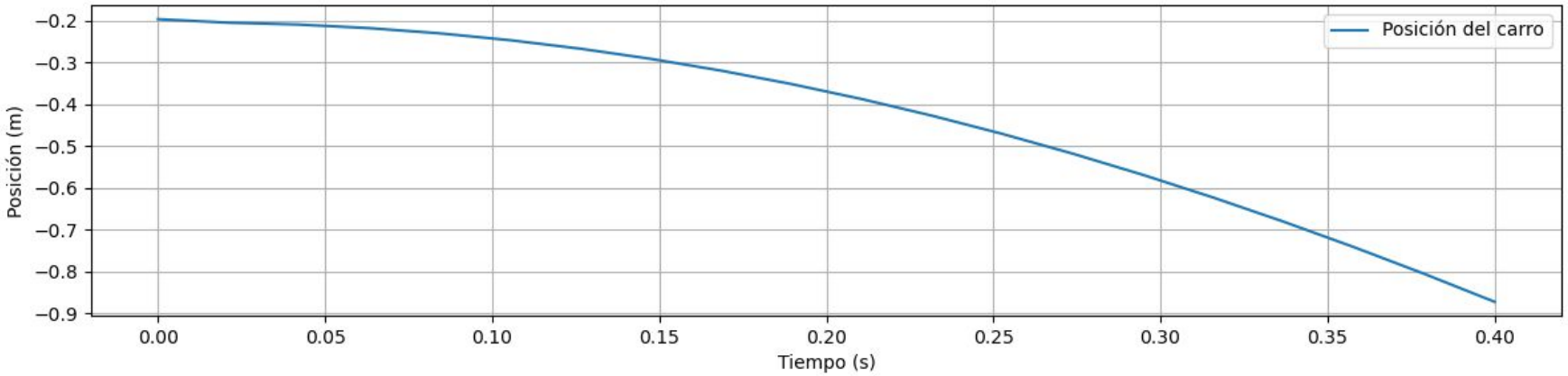
Experimentación

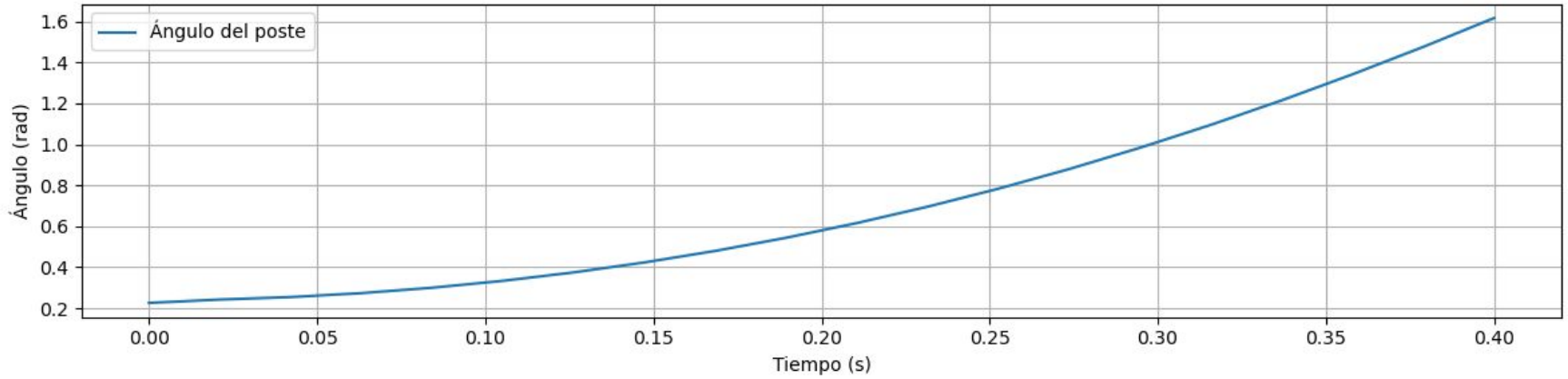
3. Algoritmo Sarsa

```
Resultados para SARSA
Alpha = 0.8
Gamma = 0.99
Epsilon = 1.0
Epsilon Decay = 0.995
Episodios = 10000
Pasos x episodio = 5000
Episodio 856, Estado final: (10, 10, 4, 7), Recompensa total: 1678.0
Episodio 2824, Estado final: (8, 5, 15, 16), Recompensa total: 687.0
Episodio 3210, Estado final: (10, 8, 14, 13), Recompensa total: 603.0
Tiempo usado por SARSA con 10000 episodios = 21.69 segundos
```

Experimentación

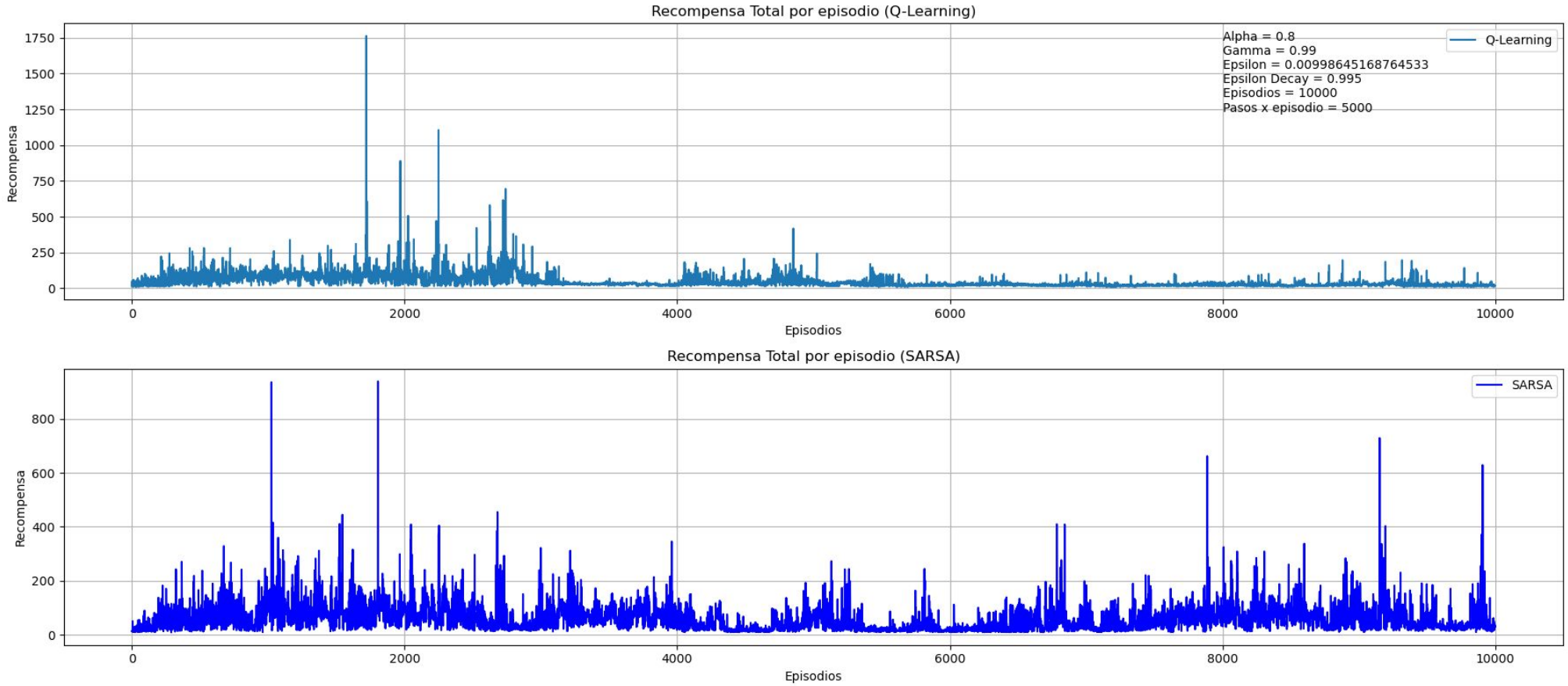
3. Algoritmo Sarsa - Simulación





Experimentación

4. Algoritmo Q-Learning vs Sarsa



Experimentación

4. Algoritmo Q-Learning vs Sarsa

```
Alpha = 0.8
Gamma = 0.99
Epsilon = 1.0
Epsilon Decay = 0.995
Episodios = 10000
Pasos x episodio = 5000
Q-Learning
Episodio 1719, Recompensa total: 1763.0
Episodio 1723, Recompensa total: 609.0
Episodio 1966, Recompensa total: 552.0
Episodio 1969, Recompensa total: 890.0
Episodio 2026, Recompensa total: 507.0
Episodio 2249, Recompensa total: 1106.0
Episodio 2625, Recompensa total: 581.0
Episodio 2723, Recompensa total: 617.0
Episodio 2741, Recompensa total: 695.0
SARSA
Episodio 1023, Recompensa total: 936.0
Episodio 1805, Recompensa total: 939.0
Episodio 7888, Recompensa total: 662.0
Episodio 9153, Recompensa total: 729.0
Episodio 9908, Recompensa total: 629.0
Tiempo Q-Learning = 21.115911960601807 segundos
Tiempo SARSA     = 22.50887107849121 segundos
```

Resultados

Ejecución Q-Learning y SARSA (página 27)

Observación	Q-Learning	SARSA
Tiempo (segundos)	21.11	22.50
Cantidad de éxitos	9	5

7. Conclusiones

Conclusiones

- Se iniciaron las experimentaciones con Alpha = 0.1 para Q-Learning y SARSA, con 10000 episodios, con los cuáles no se podían superar la recompensa de 50 puntos por episodio .
- Se cambio el valor del coeficiente de aprendizaje Alpha = 0.8 lográndose encontrar recompensas mayores a 500 puntos por episodio, de acuerdo a gymnasium, se considera resuelto el sistema de CartPole-v1.
- Con el nuevo Alpha, en 10000 episodios se Q-Learning tiene mayores recompensas por episodio que SARSA. En ese rango de episodios se genera mejor entrenamiento para Q-Learning.
- De lo anterior, debido a la calidad de la data entrenada, se demuestra que la simulación de Q-Learning (gráfico página 23) llega al equilibrio mientras que SARSA (gráfico página 26) no permite llegar al equilibrio .

8. Bibliografía

Bibliografía

- **Gymnasium Cart-Pole Environment**:

  Cart-Pole Environment in Gymnasium

  https://gymnasium.farama.org/environments/classic_control/cart_pole/

  Gymnasium Documentation

  https://gymnasium.farama.org/
- **Aprendizaje por Refuerzo:**

  Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto.

GRACIAS . . .