# BTP Report

Himani Sharma
2022102032

September 2025

## Contents

# 1 Problem setup and design requirements

## 1.1 Problem Understanding

The system consists of an inverted pendulum mounted to a motorized cart.

**Features of this system:**

1. It is unstable without control, i.e., the pendulum will simply fall over if the cart isn't moved to balance it.
2. The dynamics of the system are nonlinear.

**Objective:**
The objective of the control system is to balance the inverted pendulum by applying a force to the cart that the pendulum is attached to.

**Assumption:**
We consider a two-dimensional problem where the pendulum is constrained to move in the vertical plane. For this system:

- The control input is the force $F$ that moves the cart horizontally.
- The outputs are the angular position of the pendulum $\theta$ and the horizontal position of the cart $x$.

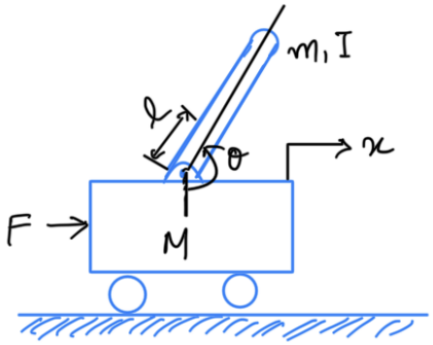


Figure 1: Inverted Pendulum

## 1.2 System Parameters

For this example, let's assume the following quantities:

- $M$ Mass of the cart 0.5 kg
- $m$ Mass of the pendulum 0.2 kg

- $b$ Coefficient of friction for cart $0.1\,\mathrm{N/m/sec}$
- $l$ Length to pendulum center of mass $0.3\,\mathrm{m}$
- $I$ Mass moment of inertia of the pendulum $0.006\,\mathrm{kg \cdot m^2}$
- $F$ Force applied to the cart
- $x$ Cart position coordinate
- $\theta$ Pendulum angle from vertical (down)

## 1.3 Design Requirements

We will design a controller to restore the pendulum to a vertically upward position after it has experienced an impulsive "bump" to the cart. Specifically, the design criteria are that the pendulum return to its upright position within 5 seconds and that the pendulum never move more than 0.05 radians away from vertical after being disturbed by an impulse of magnitude 1 Nsec. The pendulum will initially begin in the vertically upward equilibrium, $\theta = \pi$.

### 1.3.1 SISO Design Requirements

- Settling time for $\theta$ of less than 5 seconds
- Pendulum angle $\theta$ never more than 0.05 radians from the vertical

In our case, the inverted pendulum system is single-input, multi-output (SIMO). Therefore, for the state-space section of the inverted pendulum, we will attempt to control both the pendulum's angle and the cart's position.
For this example, We will command a 0.2-meter step in the cart's desired position.

### 1.3.2 State-Space Design Requirements

- Settling time for $x$ and $\theta$ of less than 5 seconds
- Rise time for $x$ of less than 0.5 seconds
- Pendulum angle $\theta$ never more than $20^\circ$ (0.35 radians) from the vertical
- Steady-state error of less than 2% for $x$ and $\theta$

# 2 Force analysis and system equations

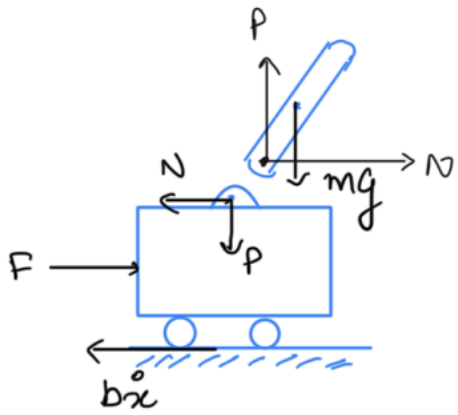


Figure 2: Forces on Inverted Pendulum

## 2.1 Equations of Motion

### 2.1.1 Pendulum

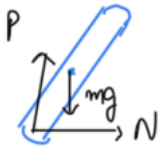


Figure 3: Pendulum

The coordinates of the center of mass are given by:

$$(x_c, y_c) \equiv (x + l\sin\theta,\ -l\cos\theta) \tag{1}$$

Velocities:

$$(v_x, v_y) \equiv \left(\dot{x} + l\cos\theta\,\dot{\theta},\ l\sin\theta\,\dot{\theta}\right) \tag{2}$$

Accelerations:

$$(a_x, a_y) = \left(\ddot{x} - l\sin\theta\,\dot{\theta}^2 + l\cos\theta\,\ddot{\theta},\ l\cos\theta\,\dot{\theta}^2 + l\sin\theta\,\ddot{\theta}\right) \tag{3}$$

Force equations for the pendulum:

$$N = m\ddot{x} \tag{4}$$

$$P - mg = m\ddot{y} \tag{5}$$

Substituting accelerations:

$$N = m\left(\ddot{x} - l\sin\theta\,\dot{\theta}^2 + l\cos\theta\,\ddot{\theta}\right) \tag{6}$$

$$P - mg = m\left(l\cos\theta\,\dot{\theta}^2 + l\sin\theta\,\ddot{\theta}\right) \tag{7}$$

### 2.1.2 Cart

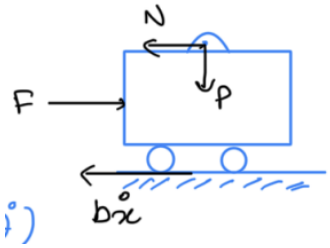


Figure 4: Cart

Force balance on the cart:

$$F - N - b\ddot{x} = M\ddot{x} \tag{8}$$

Substituting $N$:

$$M\ddot{x} = F - b\ddot{x} - m\left(\ddot{x} - l\sin\dot{\theta}\,\dot{\theta}^2 + l\cos\theta\,\ddot{\theta}\right) \tag{9}$$

Rearranging:

$$(M+m)\ddot{x} + b\ddot{x} = F + m\left(l\sin\theta\,\dot{\theta}^2 - l\cos\theta\,\ddot{\theta}\right) \tag{10}$$

Thus, the final cart equation is:

$$F = (M+m)\ddot{x} + b\dot{x} + m\left(l\cos\theta\,\ddot{\theta} - l\sin\theta\,\dot{\theta}^2\right) \tag{11}$$

### 2.1.3 Rotation (Moment) Equation

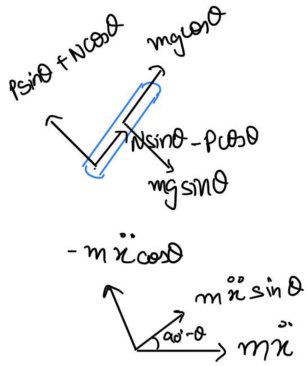


Figure 5: Rotation Forces

According to Newton–Euler formulation:

$$\text{Torque} = \text{rate of change of angular momentum about pivot} \tag{12}$$

Thus, the torque balance is:

$$\sum \tau_{\text{pivot}} = I_p \ddot{\theta} \tag{13}$$

where the moment of inertia about the pivot is obtained using the parallel axis theorem:

$$I_p = I + ml^2 \tag{14}$$

Torque balance about the pivot:

$$P \sin\theta + N \cos\theta - mg \sin\theta = ml\ddot{\theta} + m\ddot{x} \cos\theta \tag{15}$$

Moment about the center of mass (COM):

$$-(P \sin\theta)l - (N \cos\theta)l = I\ddot{\theta} \tag{16}$$

Combining both and eliminating $P, N$:

$$-mgl \sin\theta = ml\ddot{\theta} + m\ddot{x}l \cos\theta + I\ddot{\theta} \tag{17}$$

Finally, the compact form is:

$$\boxed{\left(I + ml^2\right)\ddot{\theta} + mgl \sin\theta = -m\ddot{x}\left(l \cos\theta\right)} \tag{18}$$

## 2.2 Linearization of the System

Since the analysis and control design techniques are only be done here for linear systems, this set of equations needs to be linearized. About the vertically upward equilibrium position, $\theta = \pi$, and assuming that the system stays within a small neighborhood of this equilibrium, which is valid since deviation less than $20^\circ$ from the vertically upward position is desirable.

Let $\phi$ represent the deviation of the pendulum's position from equilibrium, that is, $\theta = \pi + \phi$. Again presuming a small deviation ($\phi$) from equilibrium, we can use the following small angle approximations of the nonlinear functions in our system equations:

$$\cos\theta = \cos(\pi + \phi) \approx -1 \tag{19}$$

$$\sin\theta = \sin(\pi + \phi) \approx -\phi \tag{20}$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0 \tag{21}$$

After substituting the above approximations into our nonlinear governing equations, we arrive at the two linearized equations of motion. Note $u$ has been substituted for the input $F$.

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \tag{22}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \tag{23}$$

## 2.3 Transfer Function

To obtain the transfer functions, we must first take the Laplace transform of the system equations assuming zero initial conditions.

$$(I + ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2 \tag{24}$$

$$(M + m)X(s)s^2 + bX(s)s - ml\Phi(s)s^2 = U(s) \tag{25}$$

Recall that a transfer function represents the relationship between a single input and a single output at a time. To find our first transfer function for the output $\Phi(s)$ and an input of $U(s)$ we need to eliminate $X(s)$ from the above equations. Solve the first equation for $X(s)$:

$$X(s) = \left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s) \tag{26}$$

Then substitute the above into the second equation:

$$(M + m)\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s^2 + b\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s - ml\Phi(s)s^2 = U(s) \tag{27}$$

Rearranging, the transfer function is then the following:

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \tag{28}$$

where,

$$q = \left[(M + m)(I + ml^2) - (ml)^2\right] \tag{29}$$

From the transfer function above it can be seen that there is both a pole and a zero at the origin. These can be canceled and the transfer function becomes the following:

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \qquad \left[\frac{\text{rad}}{\text{N}}\right] \tag{30}$$

Second, the transfer function with the cart position $X(s)$ as the output can be derived in a similar manner to arrive at the following:

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \qquad \left[\frac{\text{m}}{\text{N}}\right] \tag{31}$$

## 2.4 State-Space Representation

The linearized equations of motion from above can also be represented in state-space form .

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Ml^2} & \frac{m^2gl^2}{I(M+m)+Ml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Ml^2} & \frac{mgl(M+m)}{I(M+m)+Ml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Ml^2} \\ 0 \\ \frac{ml}{I(M+m)+Ml^2} \end{bmatrix} u \tag{32}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \tag{33}$$

The $C$ matrix has 2 rows because both the cart's position and the pendulum's position are part of the output. Specifically, the cart's position is the first element of the output $\mathbf{y}$ and the pendulum's deviation from its equilibrium position is the second element of $\mathbf{y}$.

# 3 MATLAB representation

## 3.1 Tranfer Function

We can represent the transfer functions derived above for the inverted pendulum system within MATLAB employing the following commands.

From input $u$ to output $x$ :

$$P_{cart}(s) = \frac{4.182 \times 10^{-6}s^2 - 1.025 \times 10^{-4}}{2.3 \times 10^{-6}s^4 + 4.182 \times 10^{-7}s^3 - 7.172 \times 10^{-5}s^2 - 1.025 \times 10^{-5}s}$$

From input $u$ to output $\phi$ :

$$P_{pend}(s) = \frac{1.045 \times 10^{-5}s}{2.3 \times 10^{-6}s^3 + 4.182 \times 10^{-7}s^2 - 7.172 \times 10^{-5}s - 1.025 \times 10^{-5}}$$

Continuous-time transfer function.

## 3.2 State -Space

We can also represent the system using the state-space equations. The following additional MATLAB commands create a state-space model of the inverted pendulum and produce the output shown below when run in the MATLAB command window.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.1818 & 2.673 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.4545 & 31.18 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1.818 \\ 0 \\ 4.545 \end{bmatrix}, \tag{34}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{35}$$

# 4 Data Generation Analysis using Gymnasium

This work uses the `InvertedPendulum-v5` environment from the `Gymnasium` library, which is a MuJoCo-based environment. The task is to balance an inverted pendulum on a cart by applying horizontal forces.

## 4.1 Import

```
import gymnasium as gym
env = gym.make("InvertedPendulum-v5")
```

## 4.2 Action Space

The action space is continuous and represented as:

$$\text{Box}([-3.0], [3.0], (1,), \texttt{float32})$$

The agent takes a one-dimensional continuous action (force applied to the cart).

| Num | Action | Control Min | Control Max | Name | Joint | Type (Unit) |
|---|---|---|---|---|---|---|
| 0 | Force on cart | -3 | 3 | slider | slide | Force (N) |

Table 1: Action space specification of InvertedPendulum-v5.

## 4.3 Observation Space

The observation space is a 4-dimensional vector:

$$\text{Box}(-\infty, \infty, (4,), \texttt{float64})$$

It consists of cart position/velocity and pole angle/angular velocity.

| Num | Observation | Min | Max | Name | Joint | Type (Unit) |
|---|---|---|---|---|---|---|
| 0 | Cart position | $-\infty$ | $\infty$ | slider | slide | position (m) |
| 1 | Pole angle | $-\infty$ | $\infty$ | hinge | hinge | angle (rad) |
| 2 | Cart velocity | $-\infty$ | $\infty$ | slider | slide | velocity (m/s) |
| 3 | Pole angular velocity | $-\infty$ | $\infty$ | hinge | hinge | angular velocity (rad/s) |

Table 2: Observation space specification of InvertedPendulum-v5.

## 4.4 Rewards

A reward of +1 is given for each timestep that the pole remains upright (within an angle limit).

## 4.5 Starting State

The initial state (position and velocity) is sampled from a multivariate uniform continuous distribution, perturbed by random noise.

## 4.6 Episode End

- **Termination:** Episode terminates if:
  - Any state value is non-finite.
  - The absolute pole angle $|\theta| > 0.2$ radians.
- **Truncation:** Default maximum of 1000 timesteps.

## 4.7 Data Generation Code

The Python code generates data from the `InvertedPendulum-v5` environment using Gymnasium. It collects state transitions, actions, and rewards for further analysis.
Here we are considering the agent to take completely random actions:

**Main goal:** The script is for **generating datasets** the Inverted Pendulum system. It runs the simulation for 50 episodes and saves the detailed step-by-step results of each run into separate `.csv` files. These datasets serve as a **baseline performance** for a random agent, which can later be used to evaluate the performance of a "smart" agent.

### 4.7.1 How the Code Works

The script simulates an inverted pendulum by interacting with a `gym` environment and collecting data. Its workflow is as follows:

1. **Environment Setup:** The environment is created using `env = gym.make(...)` and initialized with `env.reset()`.
2. **Episode Loop:** For each episode, the pendulum is simulated until it either falls (`terminated`) or the time limit is reached (`truncated`).
3. **Step Loop:** Within each episode, random actions are generated (`env.action_space.sample()`) and applied (`env.step(action)`). The resulting next state, reward, and episode status are returned.
4. **Data Collection:** At each step, the current state, chosen action, reward, next state, and termination flag are recorded in `collected_data`. The current observation is updated accordingly.
5. **Cleanup:** After all episodes, the environment is closed using `env.close()`.

## 4.8 Analysis of Simulation Plots

The following plots were generated for ($N = 50$) numbers of episodes . Each plot provides a different perspective on the behavior of the inverted pendulum under random actions.

### 4.8.1 Histogram of Episode Durations

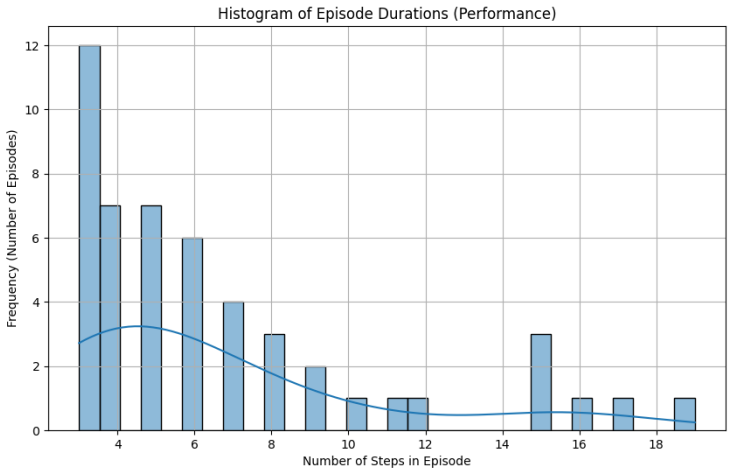


Figure 6: Histogram of episode durations for $N = 50$.

This plot serves as the main performance indicator, showing how successful the random agent was at balancing the pole.

- **X-axis**: Episode duration (number of steps).
- **Y-axis**: Frequency of episodes with that duration.
- **Interpretation**: For a random agent, the distribution is typically skewed to the left, meaning most episodes are very short. A few "lucky" runs may last longer, forming a long right tail. With larger $N$, the histogram gives a more reliable picture of performance.
- **Insight:** This confirms that the agent's performance is consistently poor, failing almost immediately in most attempts.

### 4.8.2 Phase Plot: Pole Angle vs. Angular Velocity

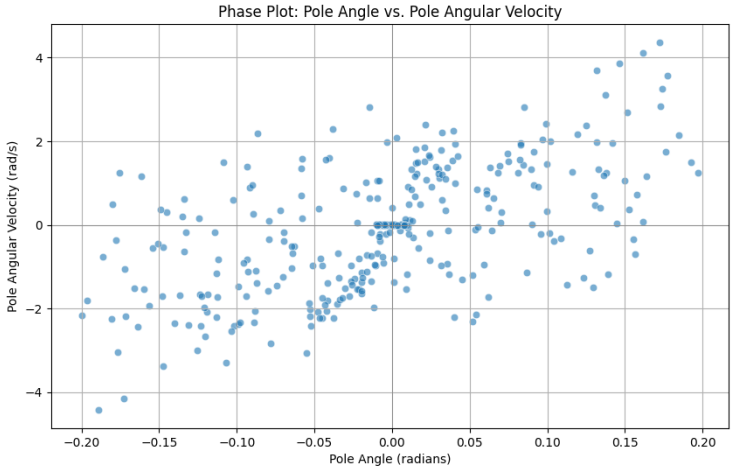


Figure 7: Phase plot: Pole angle vs. angular velocity.

This phase-space plot captures the underlying physical dynamics of the pendulum.

- **X-axis**: Pole angle (radians), where 0 is upright.
- **Y-axis**: Angular velocity of the pole (rad/s).
- **Interpretation**: Each point represents the system state at a particular timestep. The goal state is $(0, 0)$, meaning upright and motionless. For a random policy, the points typically spread outward from the center, reflecting that once the pole starts to fall, its angular velocity grows quickly, leading to failure.
- **Insight:** This illustrates the inherent instability of the inverted pendulum. Random actions cannot restore the system state back toward the stable center.

### 4.8.3 Time Series of the Longest Episode

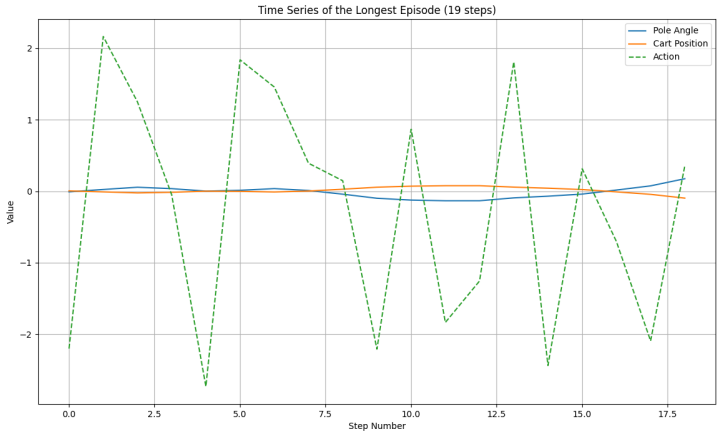


Figure 8: Time series of the longest episode.

This plot highlights the trajectory of the *most successful* episode (the one who succeeded with the maximum number of timesteps without failing).

- **X-axis**: Timestep within the episode.
- **Y-axis**: Values of system variables (cart position, pole angle, etc.).
- **Interpretation**: The plot shows how state variables evolve over time, alongside the (random) control actions. It gives insight into how actions affect the system dynamics, even if the control policy is not optimal.
- **Insight:** The pole angle stays close to zero for a short period before drifting too far, causing the episode to end. The action signal is erratic, even in this "successful" run. This shows that the agent has no control strategy—only chance kept the system balanced briefly.

### 4.8.4 Reward per Episode

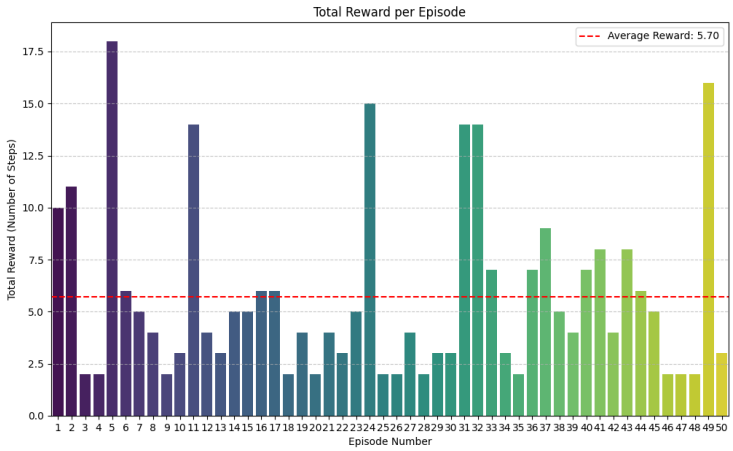


Figure 9: Total reward earned in each episode.

This bar chart summarizes the total reward earned in each episode.

- **X-axis**: Episode index.
- **Y-axis**: Total reward (equivalent to episode duration).
- **Interpretation**: With random actions, episode rewards fluctuate without any trend. Most bars are short (early failures), with a few taller bars (lucky long runs). A red dashed line indicates the average reward across episodes, which remains low for a random agent.
- **Insight:** The reward (i.e., duration) fluctuates randomly across the 50 episodes, with no upward trend. The average reward (red dashed line) is very low. This reinforces that the agent is not learning or improving. Its success is erratic and depends entirely on luck.

# 5 Conclusion and Improvements

## 5.1 Conclusion about Random Agent

The four plots paint a clear and consistent picture: a random action strategy is completely inadequate for solving the Inverted Pendulum problem. The agent exhibits no learning or strategic behavior, and its performance relies purely on chance. These results establish a quantitative baseline for failure, against which more advanced control and reinforcement learning approaches can be compared.

## 5.2 Improvements

Here we outline one of the potential improvements that can be employed to achieve meaningful control of the Inverted Pendulum.

The current random agent provides a baseline, but it cannot learn or adapt. The following improvements describe how to create a more intelligent agent that learns to balance the Inverted Pendulum effectively.

### 5.2.1 Implement a Reinforcement Learning Algorithm

The fundamental change is to replace the random action selection with a learning algorithm. Instead of guessing, the agent will have a *policy* that it improves over time. For the Inverted Pendulum problem, the recommended approach is to use a Reinforcement Learning (RL) algorithm, which is **Proximal Policy Optimization (PPO)** in this case.

PPO is a powerful, stable, and widely used RL algorithm. It works exceptionally well for control problems like the Inverted Pendulum and is a great starting point to achieve strong results.

### 5.2.2 Use a Reinforcement Learning Library

Implementing PPO from scratch is complex. Hence we are using a pre-built library that handles the low-level details, **Library: Stable Baselines3** which integrates seamlessly with Gymnasium environments and simplifies training a powerful agent.

### 5.2.3 Train the Agent

1. **Training Phase** In this phase, the agent interacts with the environment to learn a policy. The Stable Baselines3 library handles the underlying RL loop:

   - The agent interacts with the environment to collect experience.
   - It uses that experience to update its policy.
   - This process repeats for a specified number of steps (e.g., 10k, 20k), progressively improving the agent's ability to balance the pole.

2. **Evaluation Phase**

   After training, the saved trained model is used to control the environment:

   - Instead of choosing random actions, the trained model predicts the optimal action at each step.
   - The agent can now balance the pole for much longer durations, often reaching the full episode limit (e.g., 1,000 steps), demonstrating learned control.

By switching from random actions to a PPO-based policy, the agent transitions from purely chance-driven behavior to intelligent, goal-directed control.

## 5.3 Analysis of Simulation Plots

The following plots were generated for ($N = 50$) numbers of episodes. Each plot provides a different perspective on the behavior of the inverted pendulum under random actions.

### 5.3.1 Histogram of Episode Durations

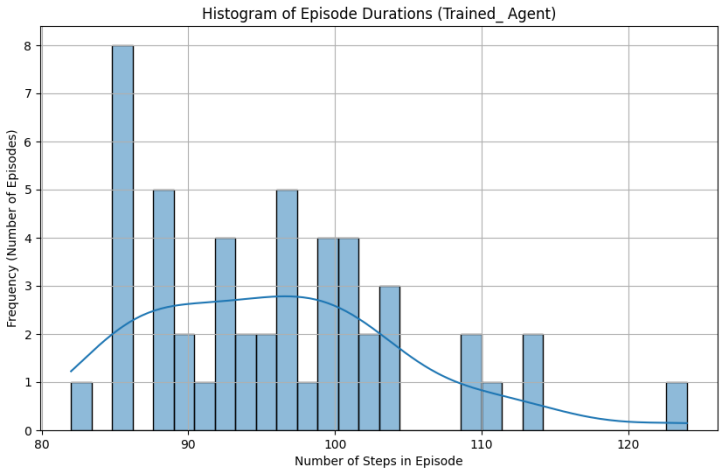


Figure 10: Histogram of episode durations for $N = 50$.

This plot serves as the main performance indicator, showing how successful the random agent was at balancing the pole.

- **Observation:** The histogram of episode durations exhibits a bimodal distribution. A good number of the episodes successfully reach the 1,000-step truncation limit, as indicated by the prominent peak on the far right. However, other cluster of episodes terminate prematurely at various shorter durations.
- **Conclusion:** This distribution signifies a policy that is successful under most tested conditions but is not yet robust enough to generalize to all possible states. The agent has clearly learned the primary stabilization task, but its strategy fails when encountering certain less-frequently experienced or more challenging state-space regions.

### 5.3.2 Phase Plot: Pole Angle vs. Angular Velocity

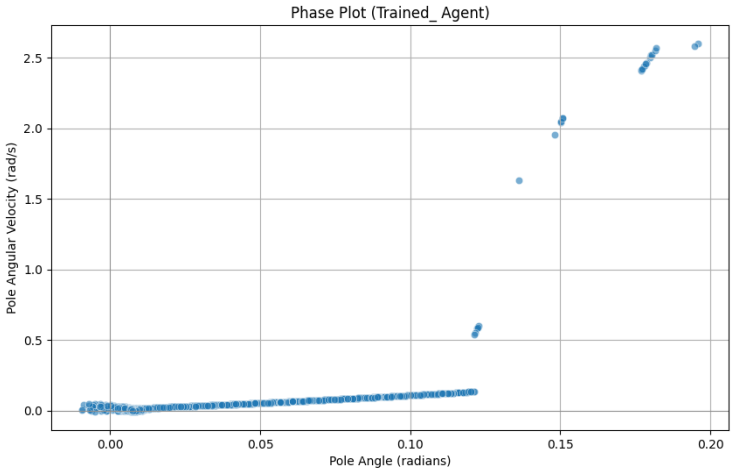


Figure 11: Phase plot: Pole angle vs. angular velocity.

This phase-space plot captures the underlying physical dynamics of the pendulum.

- **Observation:** The phase points are tightly clustered around the origin $(0, 0)$, very similar to the perfectly trained case.
- **Conclusion:** This indicates that the agent maintains the pendulum in a balanced and stable state for most of the time. The rare failures are too brief to affect this plot significantly.

### 5.3.3 Time Series of the Longest Episode

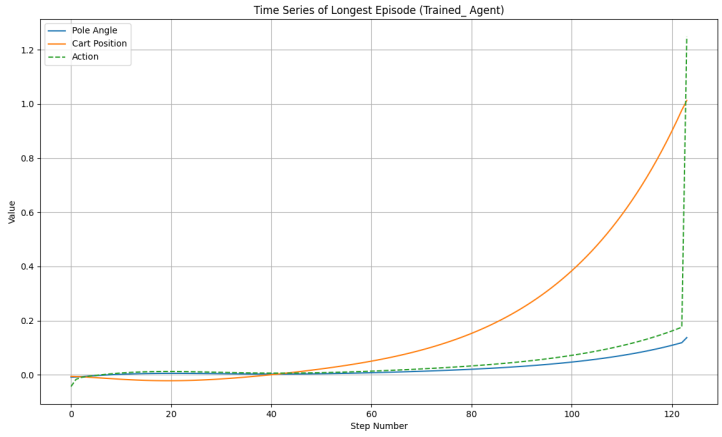


Figure 12: Time series of the longest episode.

This plot highlights the trajectory of the *most successful* episode.

- **Observation:** The 1000-step run demonstrates that the pole's angle is controlled effectively, with smooth and deliberate actions rather than random movements.
- **Conclusion:** This confirms that the agent has indeed learned the *correct control strategy*. When it succeeds, it does so with skill, not by chance.

### 5.3.4 Reward per Episode

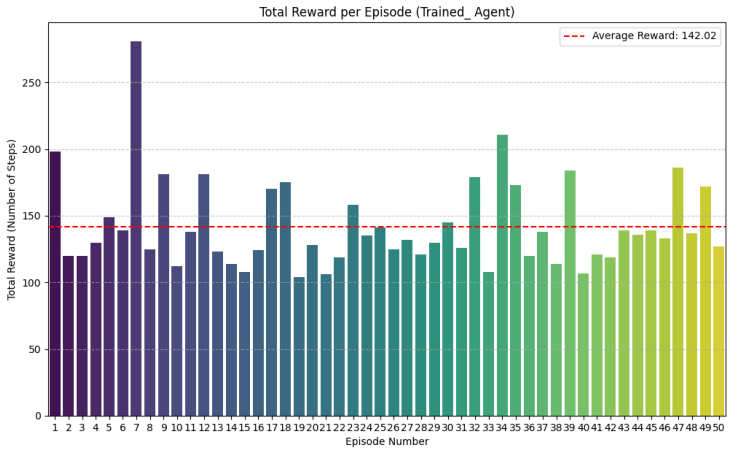


Figure 13: Total reward earned in each episode.

This bar chart summarizes the total reward earned in each episode.

- **Observation:** The bar chart displays the total reward for each of the 50 evaluation episodes. The vast majority of episodes achieve the maximum possible reward of 1000, as shown by the consistently tall bars. However, a small number of episodes terminate prematurely, resulting in lower rewards.
- **Insight:** The agent has successfully learned a highly effective control policy, transitioning from the chaotic, low-performance behavior of a random agent to a skilled, strategic one.

## 5.4 Final Conclusion

- The agent trained for 10,000 timesteps demonstrates a fundamentally different and vastly superior mode of operation compared to the baseline random policy.
- The random policy is an open-loop strategy, where actions are independent of the system state. This leads to:
  - rapid system divergence,
  - uniformly poor performance,
  - complete inability to stabilize the pendulum,
  - universally short episode durations.
- In contrast, the trained policy is a closed-loop control strategy that maps states to actions effectively. Its advantages include:
  - **Achieving Stability:** actively drives the system state towards the stable equilibrium point, unlike the random policy which accelerates divergence.
  - **Sustaining Performance:** balances the pendulum up to the maximum time limit in most trials, a feat statistically impossible for a random policy.
  - **Exhibiting Skillful Control:** actions are deliberate and corrective, opposing the system's unstable dynamics, unlike the uncorrelated and ineffective actions of the random policy.

Furthermore, increasing the training horizon, i.e., the `total_timesteps`, enhances the agent's performance. With more training iterations, the agent converges to more accurate policies, yielding greater stability, robustness, and improved control strategies compared to those obtained with fewer training steps.