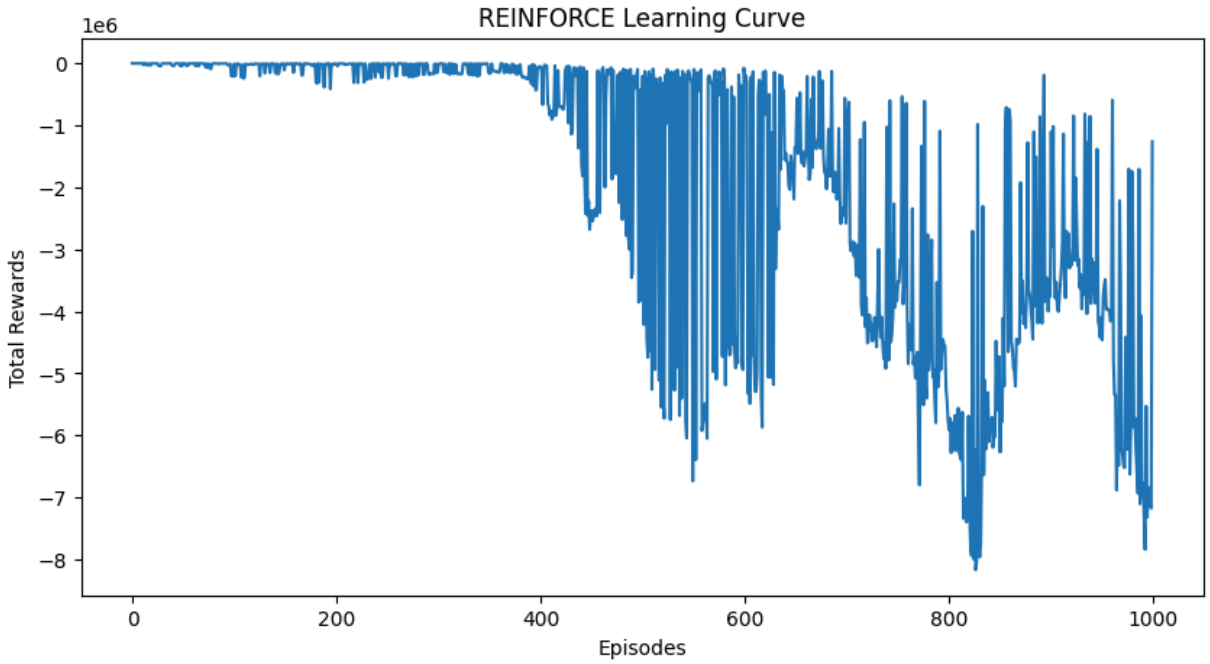
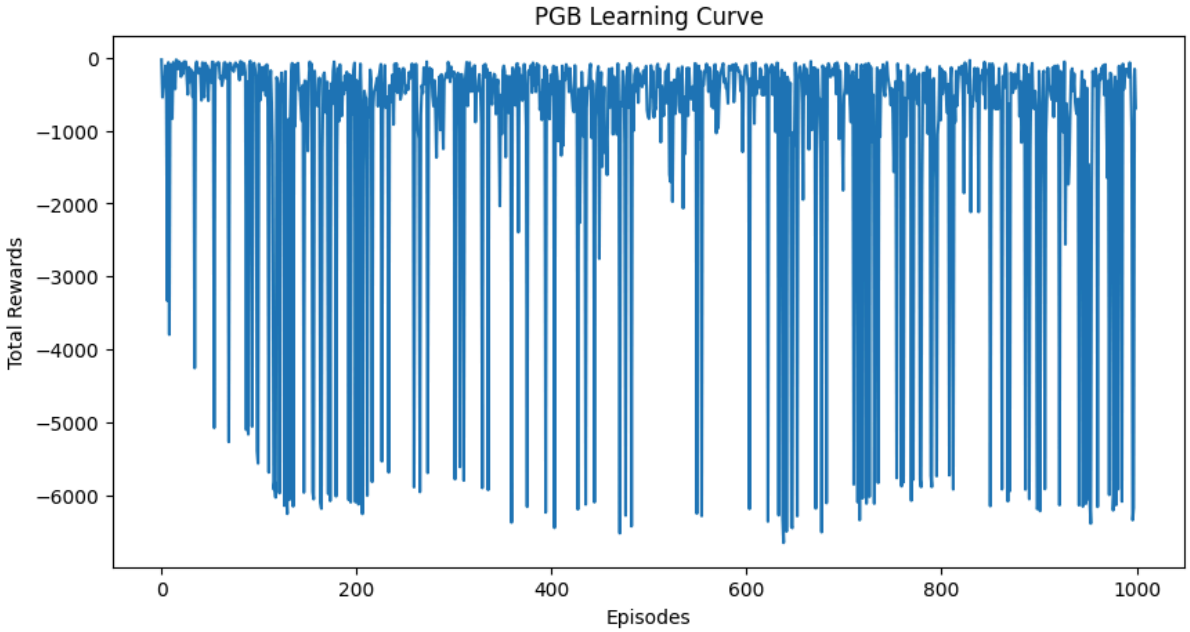
ATINDRA SHEKHAR (as5120)

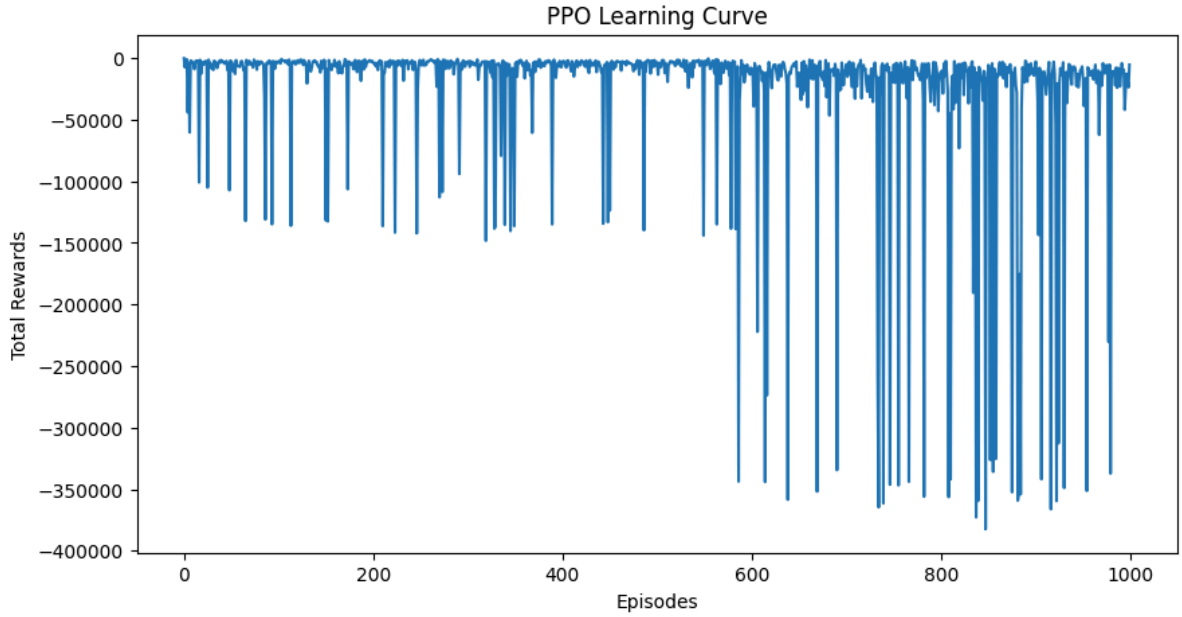
1. REINFORCE:



PGB Learning:



PPO:



2.

a. REINFORCE:
The learning curve from the REINFORCE algorithm exhibits high initial rewards followed by significant variability and a downward trend, suggesting potential instability in training. The Python implementation involves sampling actions from a normal distribution parameterized by the policy network's outputs, which are controlled using clamps to maintain stability. The training utilizes gradient descent with clipped gradients to avoid the exploding gradient problem, a technique essential for maintaining the numerical stability of updates. These measures, including clamps and clipped gradients, are crucial for maintaining the stability of the algorithm. Despite these measures, the fluctuating performance could indicate issues with either the policy network's architecture, the exploration-exploitation balance, or sensitivity to the environment's complexity.

b. PGB Learning:
The graph illustrates the learning curve of the Policy Gradient with Baseline (PGB) algorithm over 1000 episodes, displaying total rewards per episode. The rewards show significant fluctuation but are constrained within a narrower range compared to the previous REINFORCE example, suggesting a slightly more stable learning process due to the inclusion of a value network. In the provided code, two networks, policy_net and value_net, determine the action probabilities and state value estimates, respectively. The implementation clamps the mean and standard deviation outputs of the policy network to prevent numerical

instabilities, and actions are sampled from this clamped distribution. After each episode, policy and value losses are computed based on the advantage (the difference between returns and value estimates), a process that theoretically reduces variance in the policy gradient updates. This theoretical reduction should instill confidence in the algorithm's performance. Both networks' parameters are updated separately using their respective optimizers, fostering a dual optimization that aims to align the policy with both immediate and long-term returns, evident from the smoother yet still variable reward pattern in the graph.

c. PPO:

The PPO Learning Curve illustrates a pattern of total rewards per episode over 1000 episodes, characterized by significant fluctuations and a general downward trend. This downward trend, which indicates a decrease in rewards over time, is a key observation. The graph shows that while the rewards start at a relatively high value, they consistently fall to lower levels with some episodes showing large negative rewards, suggesting potential instability or inefficiencies in the policy's performance over time.
In the accompanying code, the PPO algorithm is implemented to optimize both the policy and value networks through repeated epochs of mini-batch updates. These updates are intended to refine the policy by using a clipped objective to limit the step size of policy updates, preventing drastic changes. The stability of the PPO algorithm is of utmost importance, as it is designed to improve training stability compared to methods that update policies based solely on sampled data from a single trajectory.
The collect_trajectories function gathers data by running the policy in the environment, while process_memory processes these trajectories by calculating returns and advantages, which are then normalized. During the PPO epochs, the algorithm calculates the new log probabilities and uses them to compute the surrogate loss, applying the clipping technique to ensure that the policy updates remain conservative. Despite these techniques, the visible large drops in rewards indicate the urgent need for further tuning of the hyperparameter settings, the appropriateness of the reward structure, or the network architecture. This tuning is crucial to handle the complexities of the environment more effectively.
