**4 – Q-Learning**

1. The reward function according to the reward specification is a cumulative reward -

For goal not reached at step T-

For goal reached at step T-

Setting the reward at the goal state as a relatively high positive value (instead of a 0) affects the agent. The high reward for goal state rewards a case of reaching the goal state after up to 100 actions higher than any case of not reaching the goal state in this time horizon. Also, it enables us to grant negative rewards for each state reached which is not the goal state, instead of 0 reward in previous settings. This way the agents is encouraged to reach goal in fewer steps (higher reward).

The required cumulative reward is set to . The car is required to reach the peak in steps at max. Since the cumulative reward in monotonically decreasing by 1 in each step for which the goal isn’t reached, if by the 175th step the car is not at the peak of the mountain, the cumulative reward is . Reaching the goal state at a later step will add to the reward, meaning we will not reach the target of .

1. We implemented the evaluation criterion –

Randomly selecting 10 starting states, each starting in position −0.5 (bottom of valley), and with a small uniformly distributed velocity. We ran the greedy policy with respect to the current Q function without exploration for each iteration.

The average success rate was defined as–

Before applying the learning process, the average success rate of reaching the top was 0.

1. We Implement the Q-learning algorithm and calculate the performance as defined in section (2).

The update of was implemented by the relation seen in lectures –

In our problem is linearly dependent on –

being the features of the problem. Therefore –

For the terminal state we get –

The results for running the algorithm until finding solution (finding solution being – reached performance (=success rate) = 1) are displayed below.

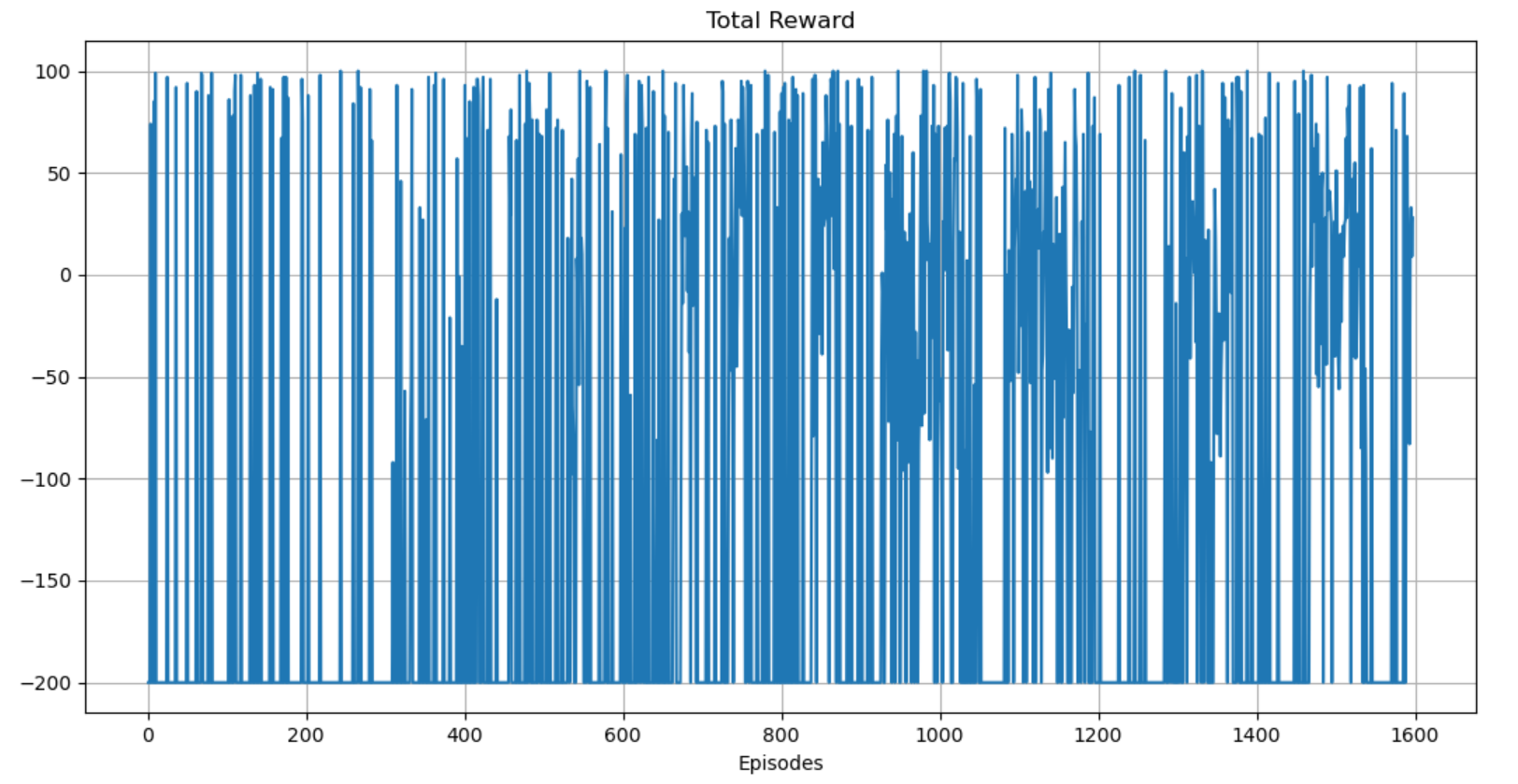
Since initial values of are randomly selected, the run time until reaching solution varied.

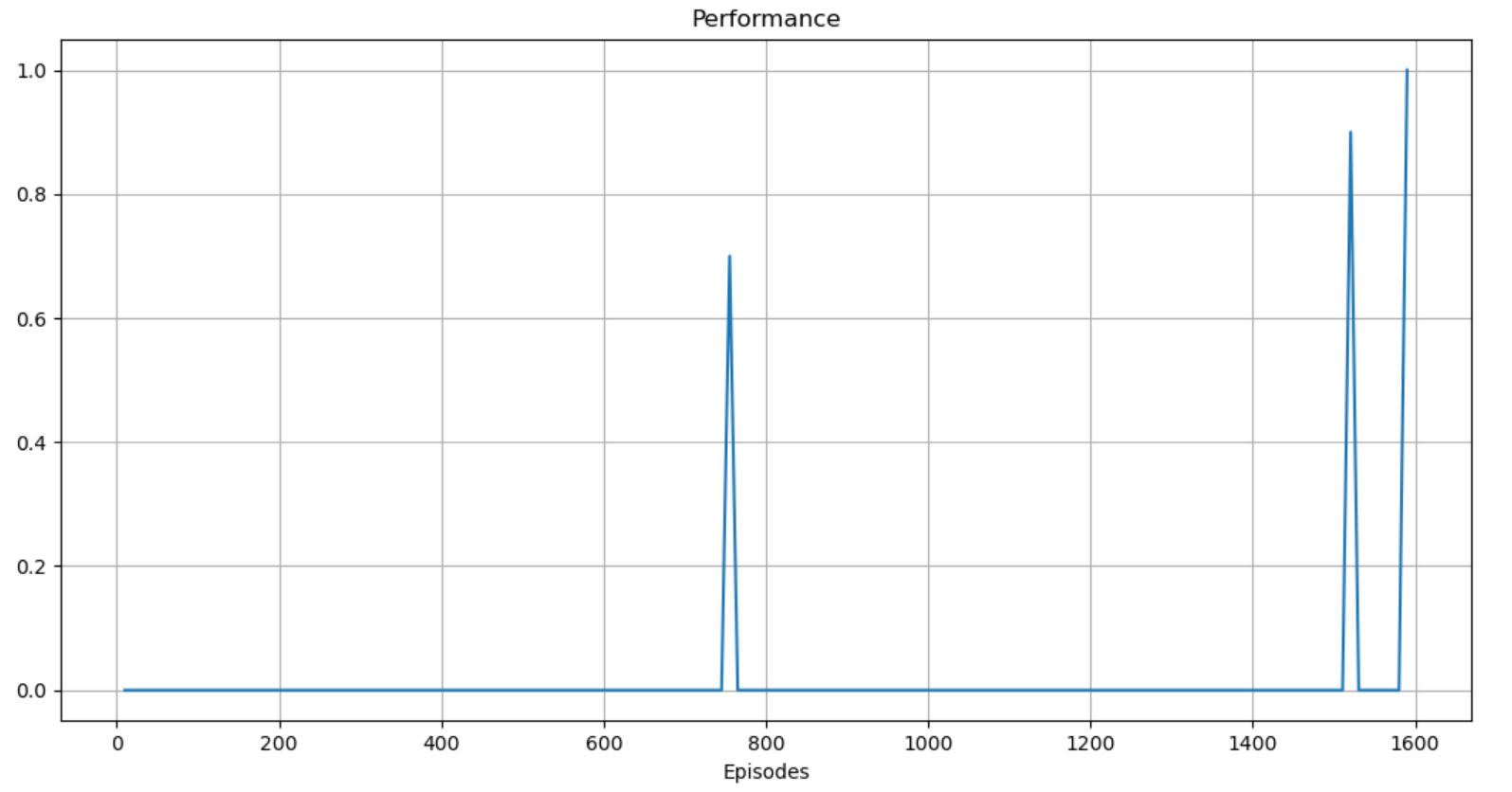
In results displayed below, solution was found after-

* First Seed = 123: 1599 episodes (first run) and after 1429 episodes (second run). Displayed for two runs to show variance of running time.
* Second Seed = 234: 309 episodes (third run).
* Third Seed = 345: 1619 episodes (forth run).

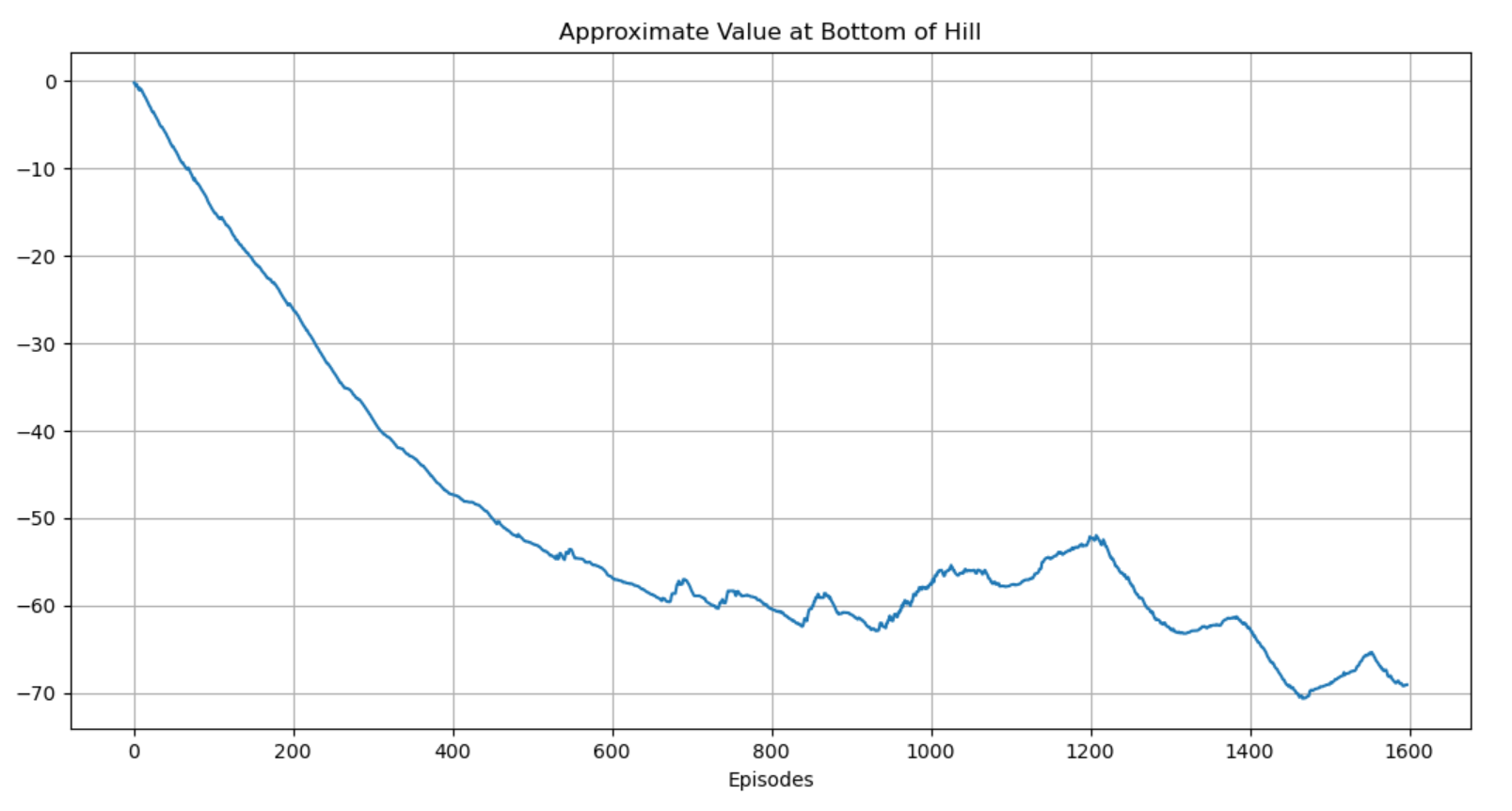
For all runs we chose -

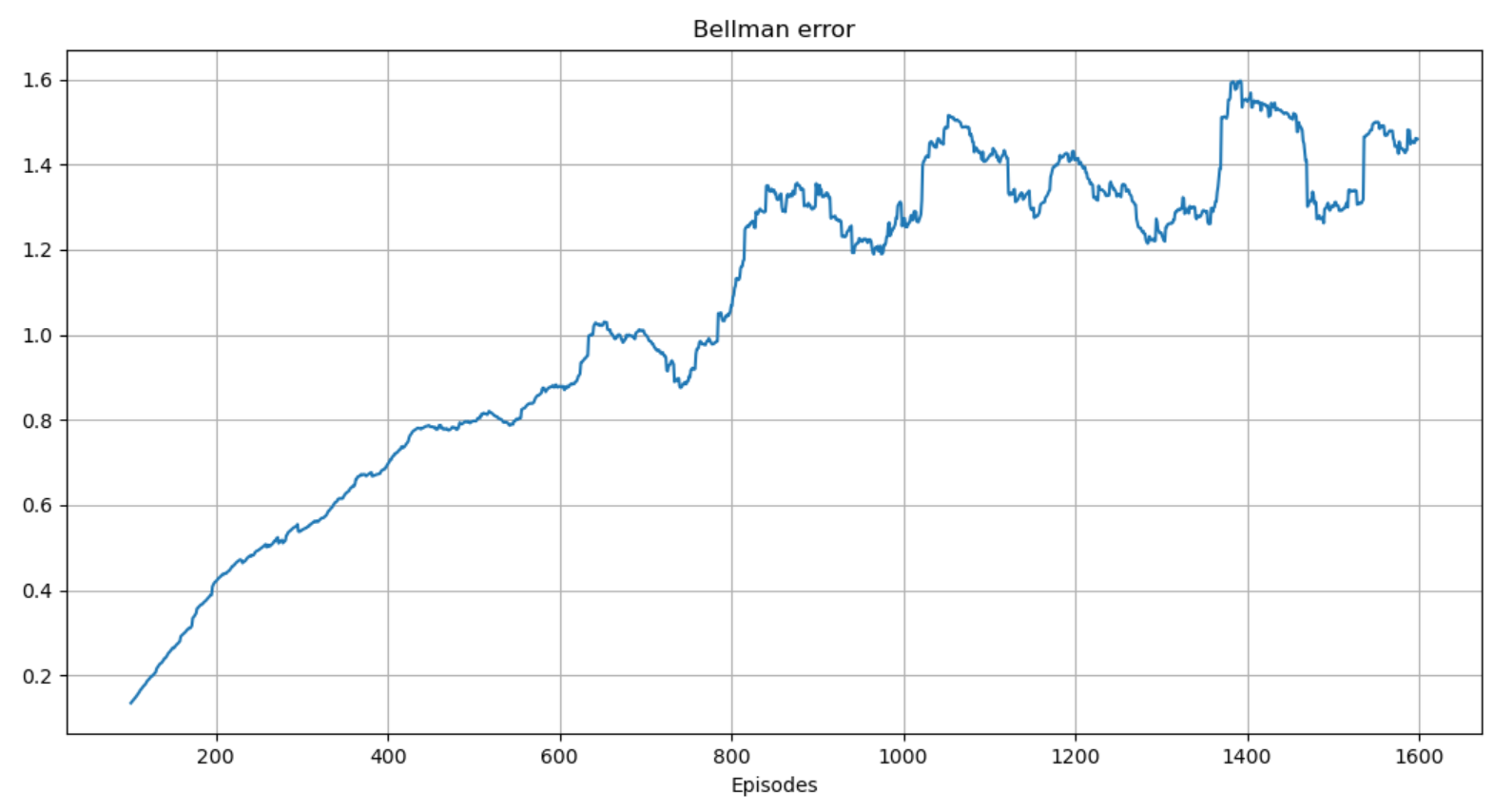
* First run – Seed = 123, 1599 episodes of training, elapsed time





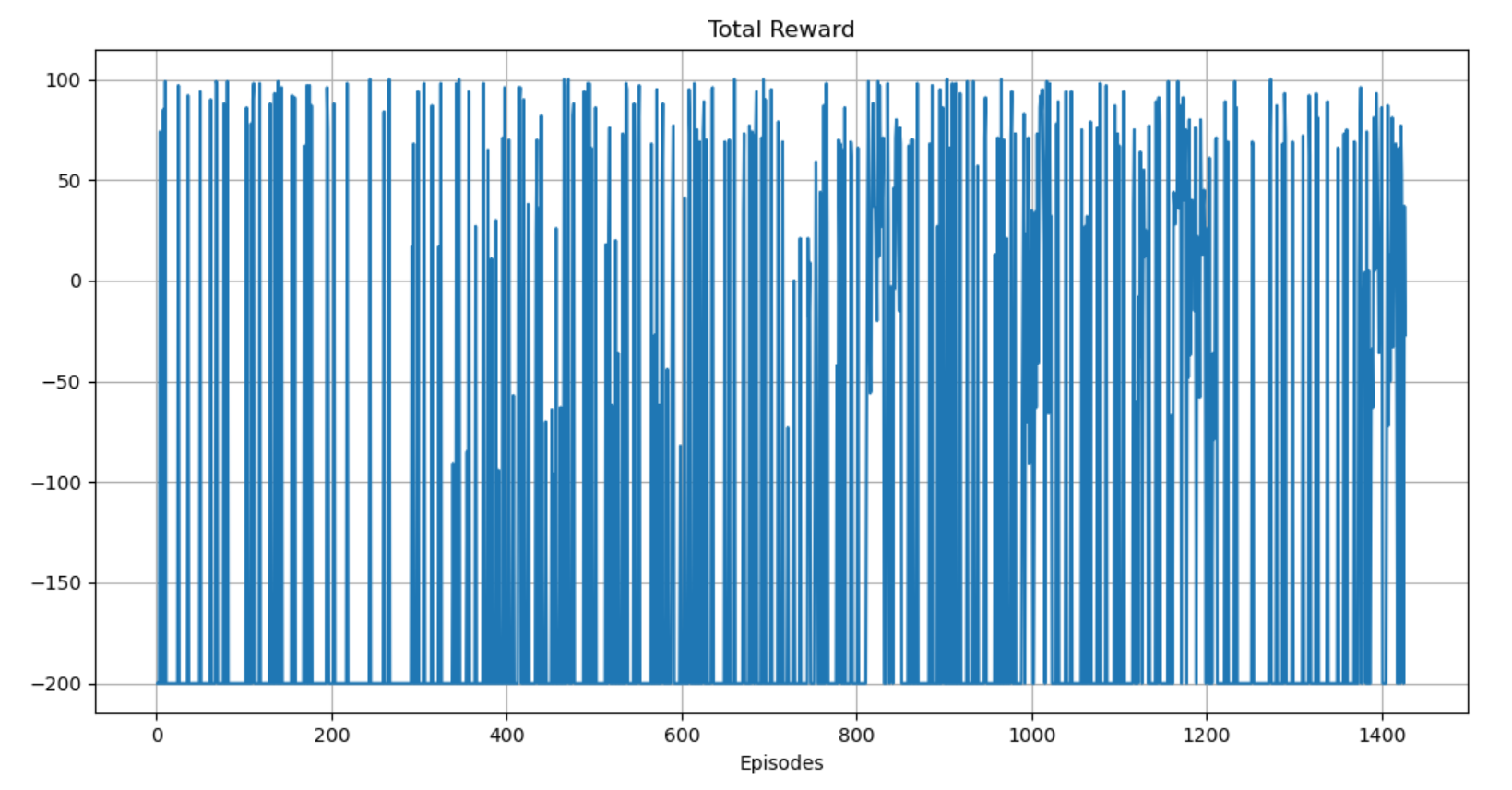
Performance was calculated every 10 episodes.

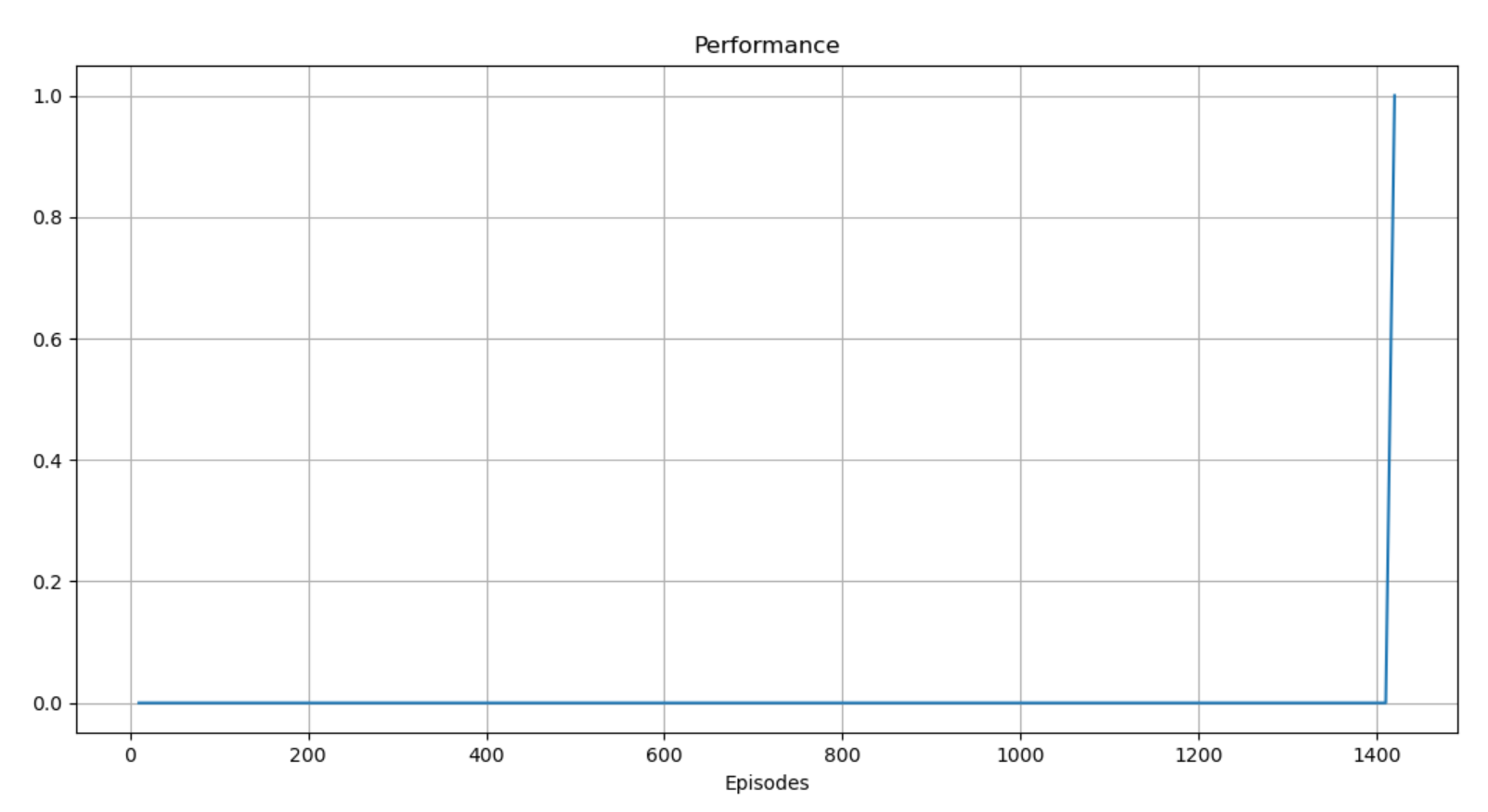




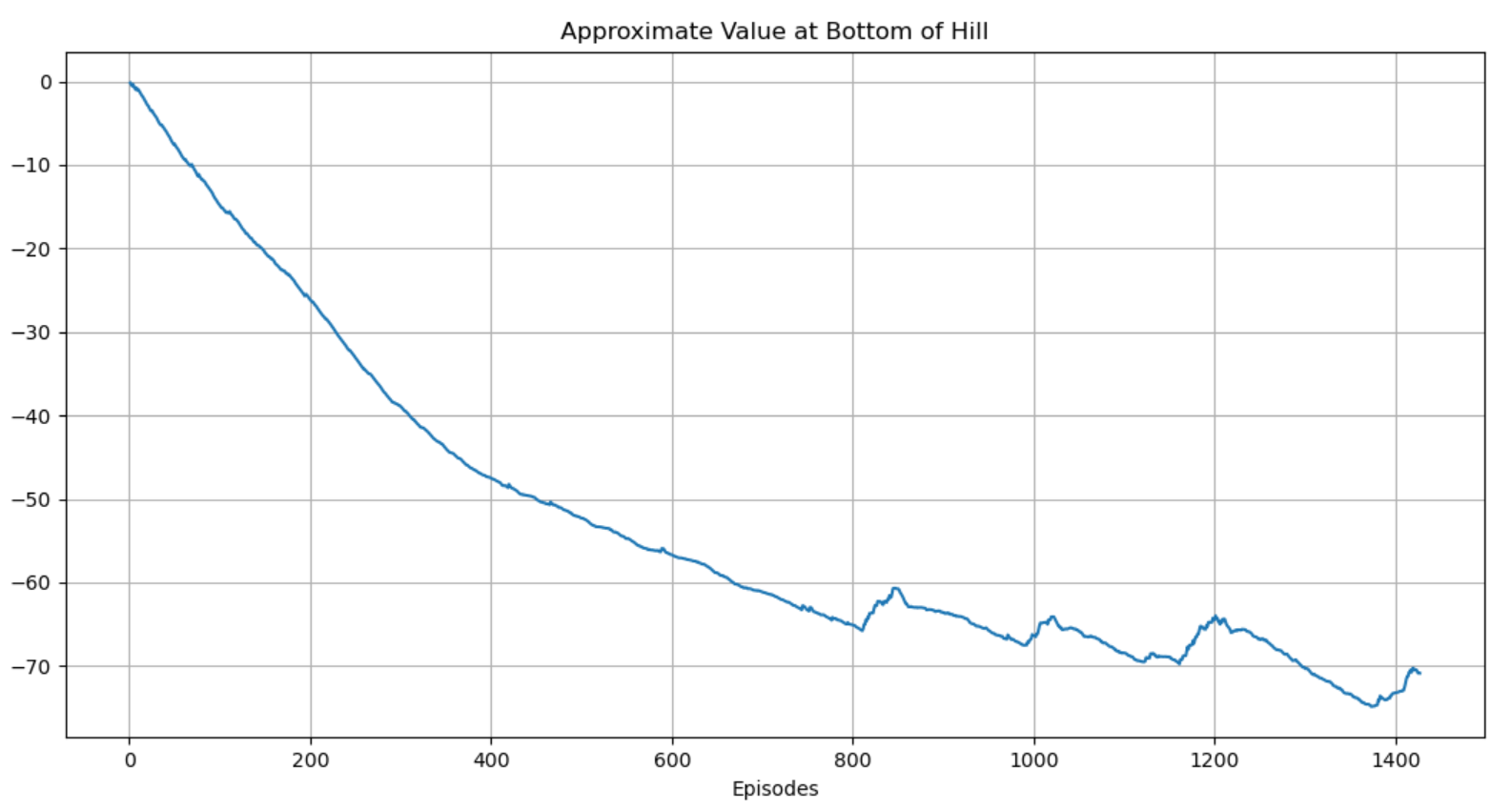
Bellman error displayed is average over 100 last episodes.

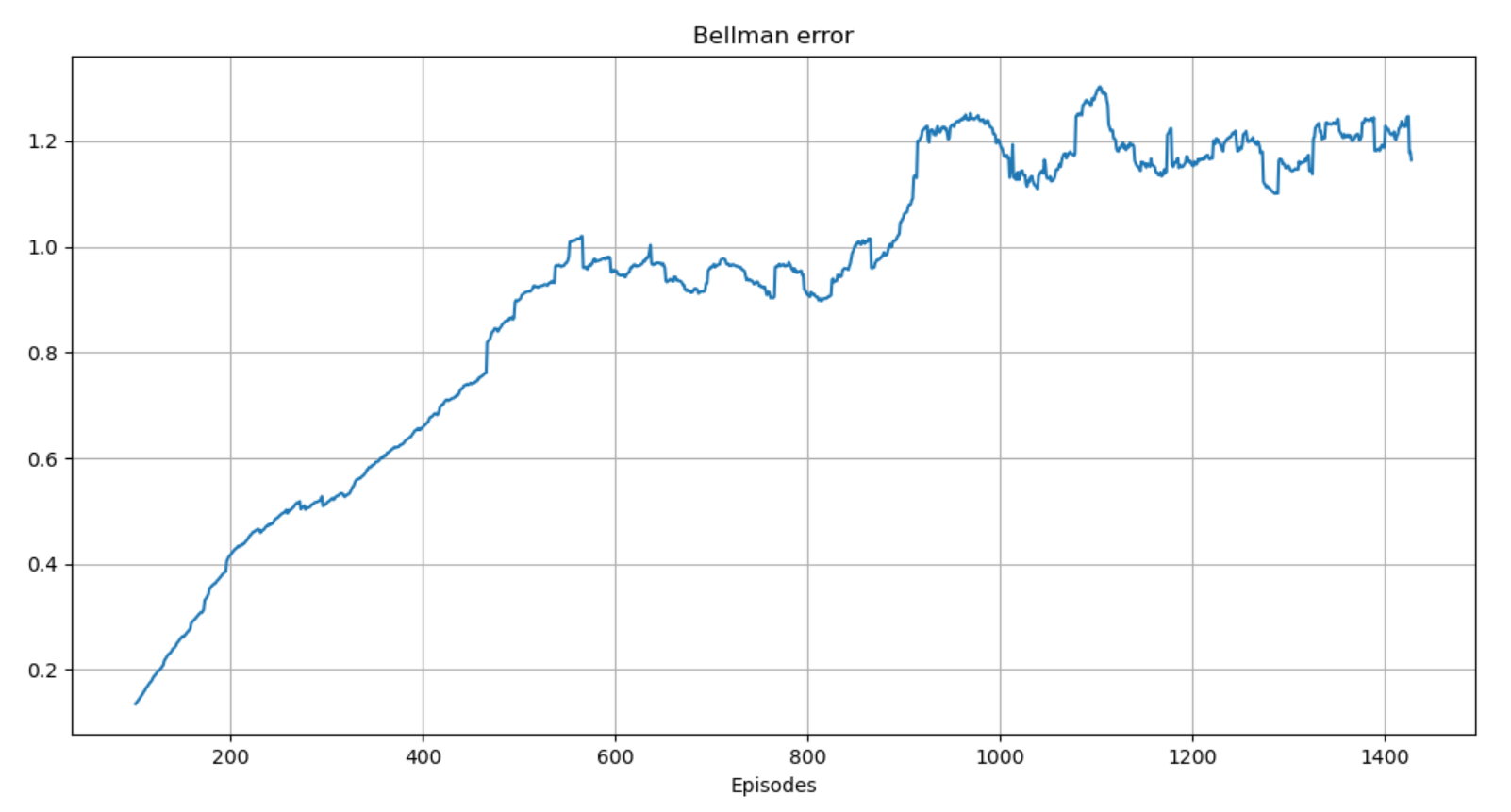
* Second run – Seed = 123, 1429 episodes of training





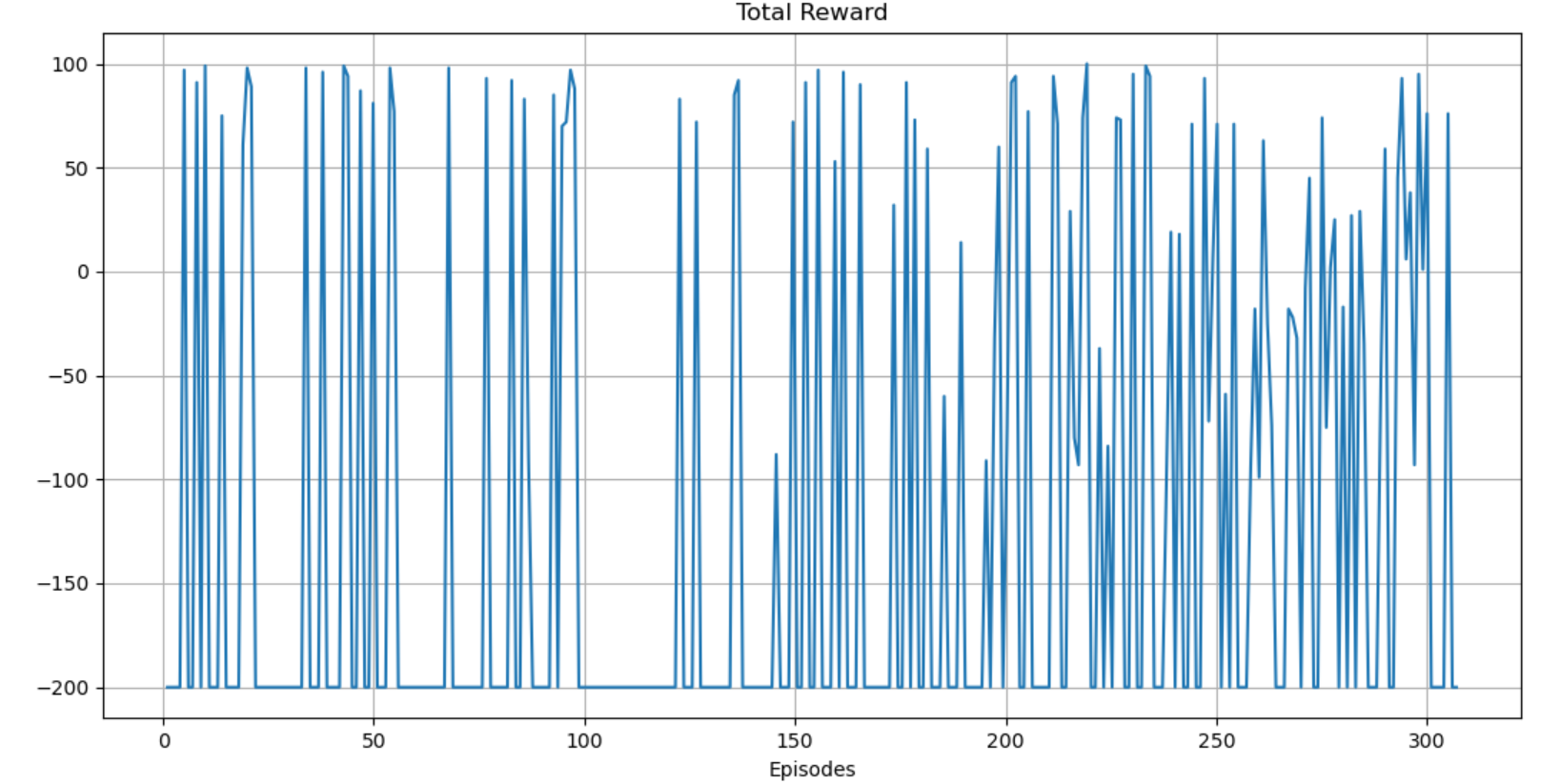
Performance was calculated every 10 episodes.

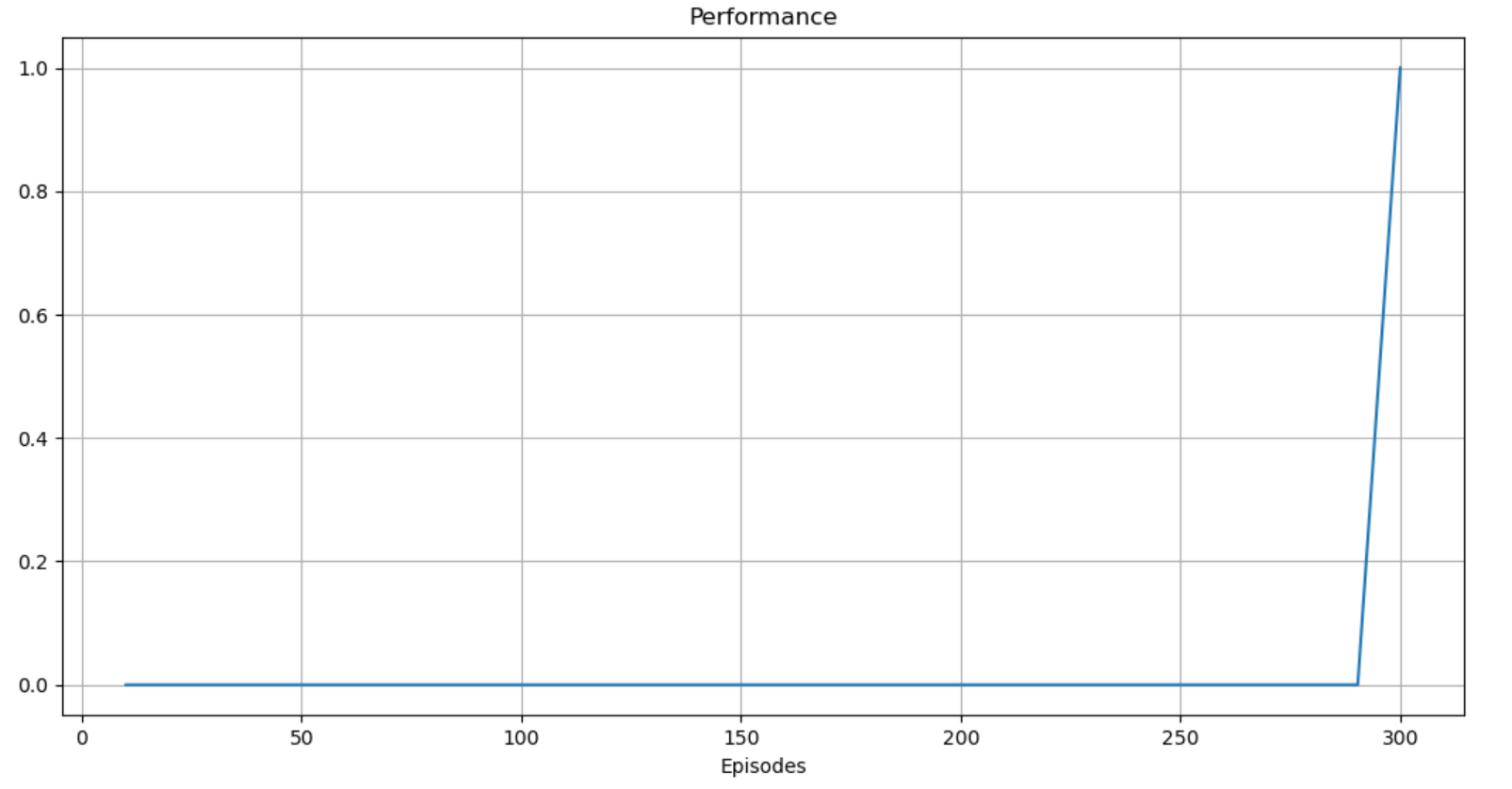




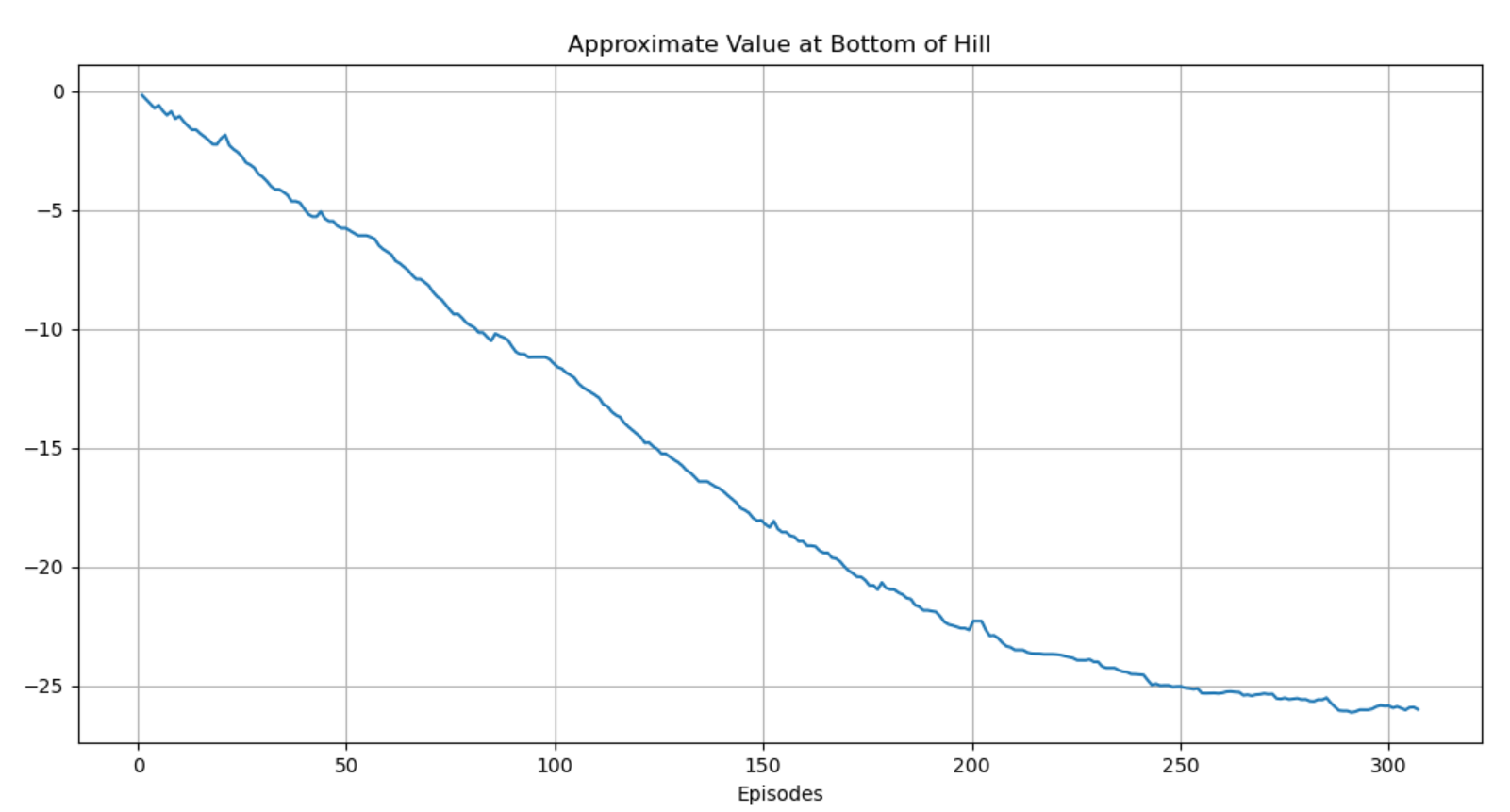
Bellman error displayed is average over 100 last episodes.

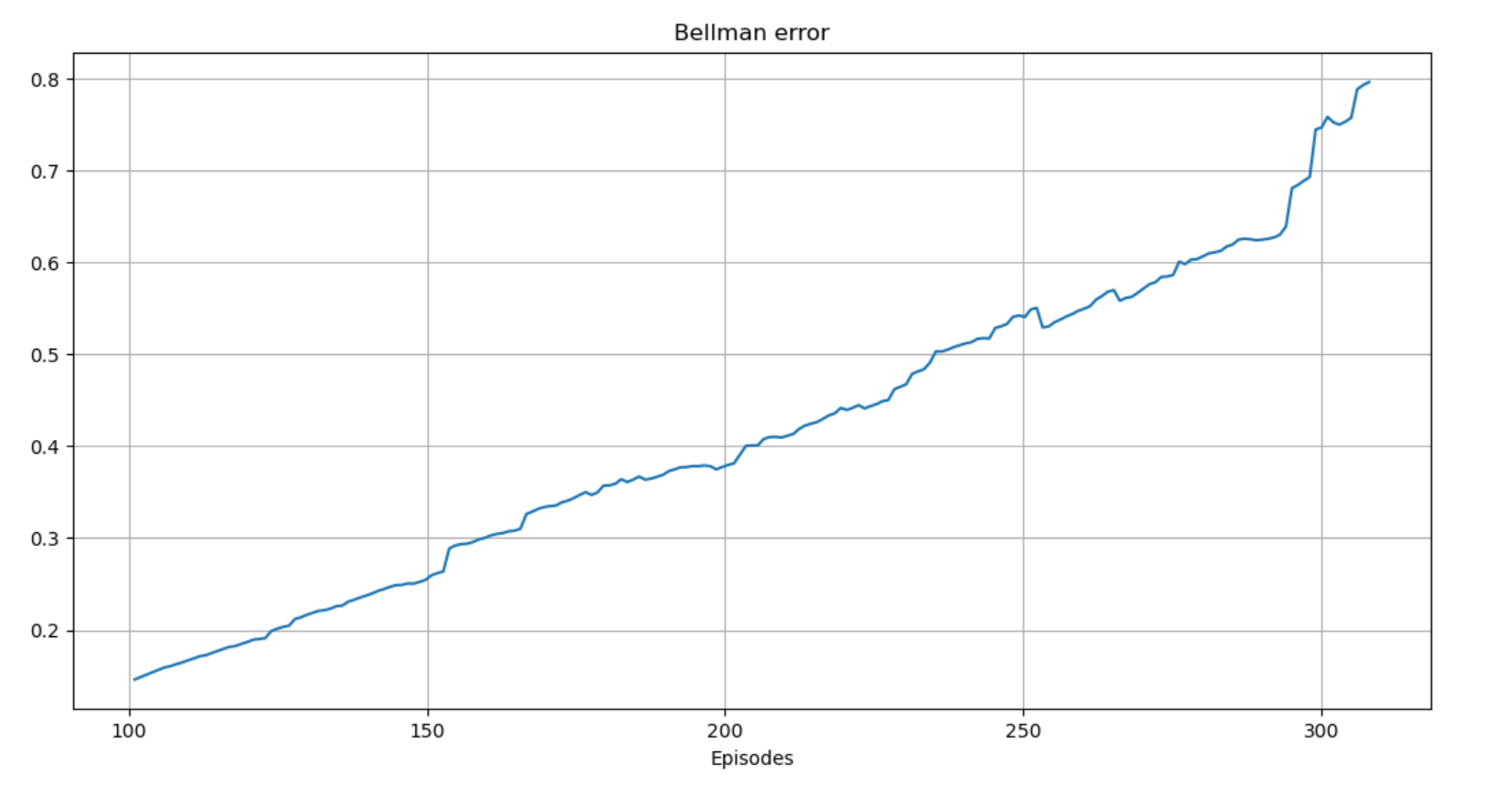
* Third run – Seed = 234, 309 episodes of training





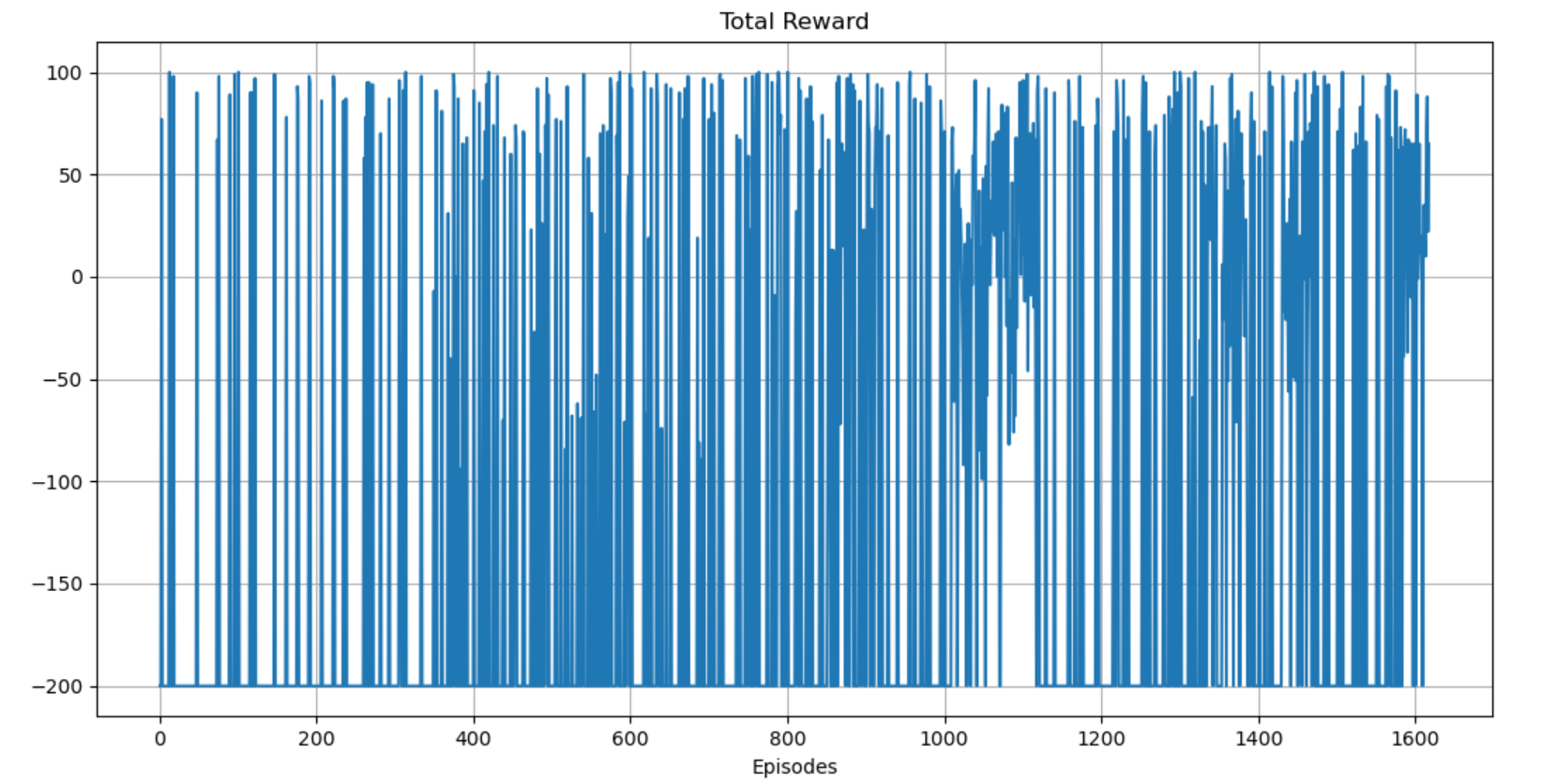
Performance was calculated every 10 episodes.

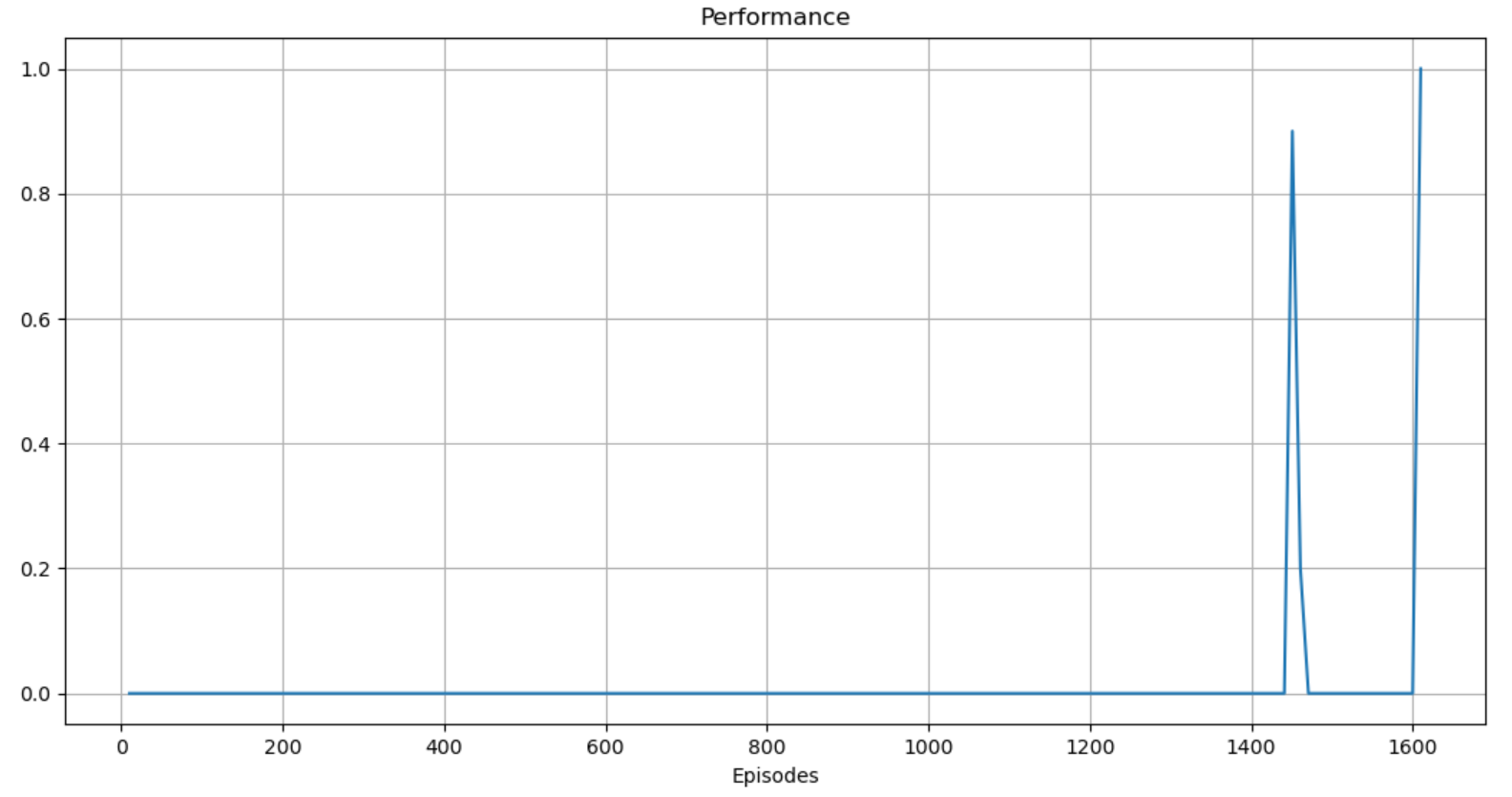




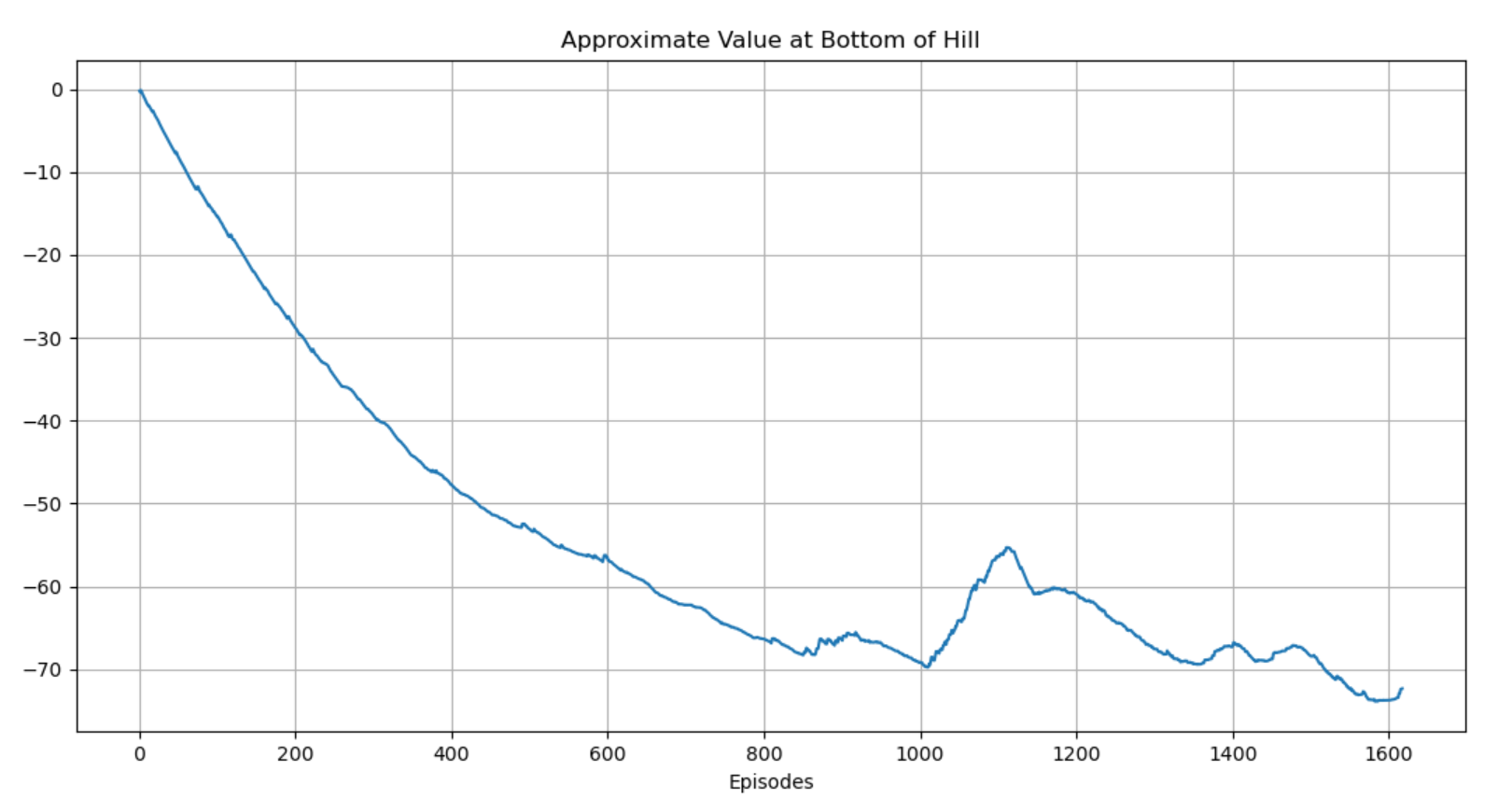
Bellman error displayed is average over 100 last episodes.

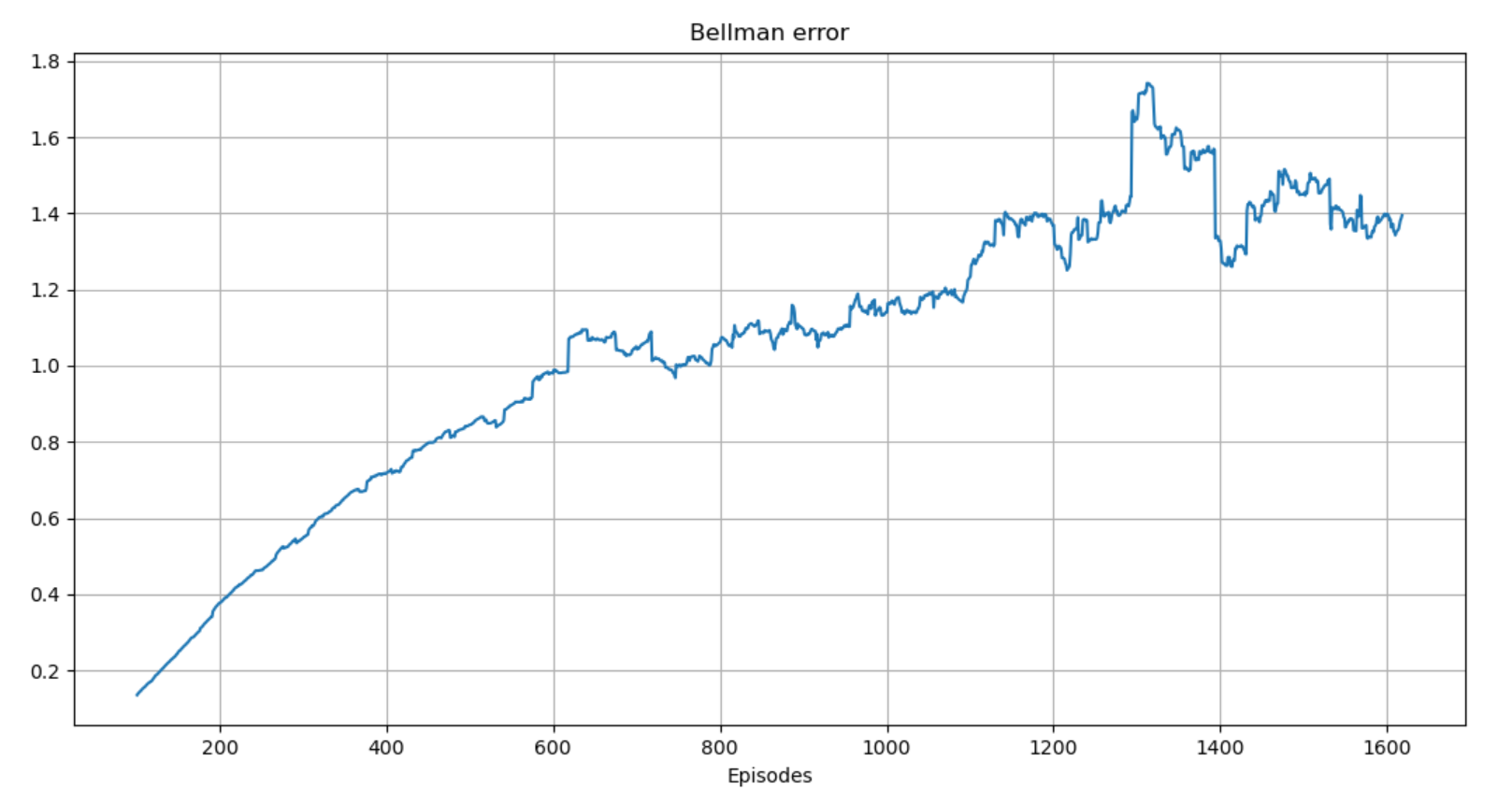
* Forth run – Seed = 345, 1619 episodes of training





Performance was calculated every 10 episodes.





Bellman error displayed is average over 100 last episodes.

The results for running the learning algorithm for 10,000 episodes (with no stopping rule prior to the 10,000th iteration for finding solution) for the same three seeds are displayed below.

* First run – Seed = 123, 10,000 episodes of training

1. We ran the greedy policy (no exploration) until solution was found (criteria for finding solution is described in previous sections (performance = 1)).