Exercise 8.1 The nonplanning method looks particularly poor in Figure 8.3 because it is a one-step method; a method using multi-step bootstrapping would do better. Do you think one of the multi-step bootstrapping methods from Chapter 7 could do as well as the Dyna method? Explain why or why not. □

An n-step method with n roughly equal to the number of planning steps Dyna-Q does would do about as well. Obvious examples of such methods are n-step Sarsa or Q-learning.

Compared to the single step method, n-step methods do a better job of updating more states when they reach the goal at the end of the first episode. The experience is used more efficiently because the update improves n states visited prior to reaching the goal.

This update is initially more efficient than an equal number of Dyna's planning steps. The random sampling of already visited state-action pairs will waste a lot of computation on trivial updates because most of the states didn't yet receive the back-propagated reward from the goal state. However, the amount of wasted compute is reduced during the subsequent episodes when more states receive an update. Later on nearly every planning step will back-propagate useful information about it's value deeper into the maze and the initial advantage of the n-step method will disappear.

As the training progresses, the key limitation of n-step methods is that their updates need direct interaction with the environment. Dyna-Q is able to back-propagate the reward it received at the end of the maze to the previous states during planning and

Exercise 8.2 Why did the Dyna agent with exploration bonus, Dyna-Q+, perform better in the first phase as well as in the second phase of the blocking and shortcut experiments? □

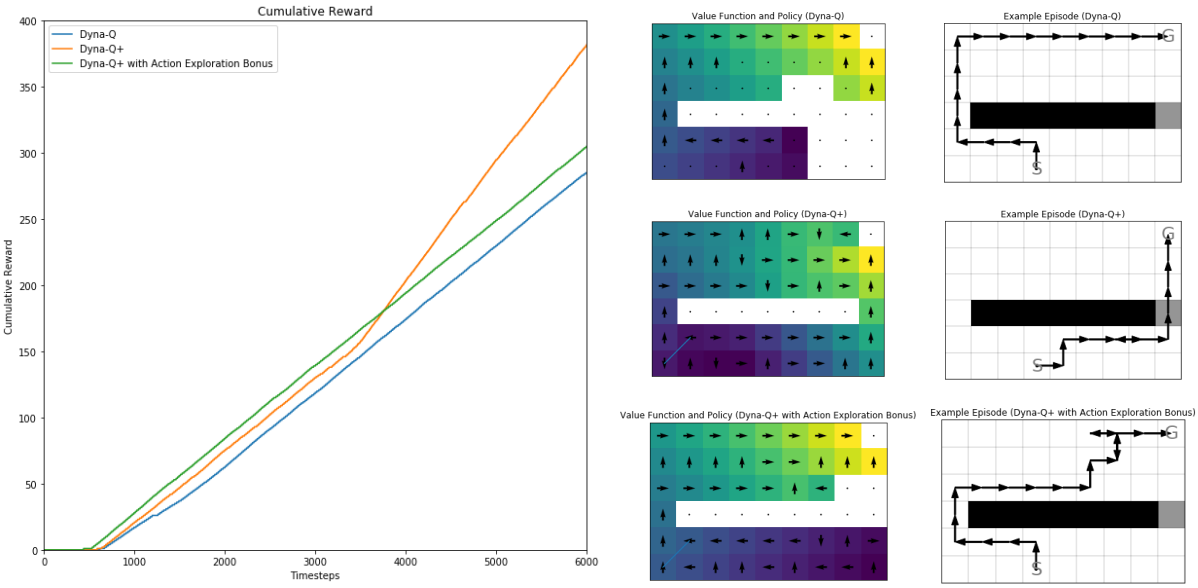
Performance in both of these phases of the experiment are dominated by the speed and efficiency of exploration. Dyna-Q+ has a much more effective exploration mechanism. The tau array logs the number of timesteps since the last visit to each state action pair. This prevents Dyna-Q+ from trying things it has tried before and at the same time keeps track of states that were not visited at all.

Dyna-Q relies only on the epsilon-greedy policy that requires much more luck and many more attempts to discover and stumble upon the reward at the goal state.

Exercise 8.3 Careful inspection of Figure 8.5 reveals that the difference between Dyna-Q+ and Dyna-Q narrowed slightly over the first part of the experiment. What is the reason for this? □

Dyna-Q+ agent is paying price for it's exploration over time. Because it explores more there are sometimes episodes that take longer and also accumulate reward slower. Dyna-Q follows epsilon-greedy policy that allows for a relatively small amount deviation or exploration. Therefore it's able to gradually reduce the gap crated by it's own slower learning during the initial discovery phase.

Exercise 8.4 (programming) The exploration bonus described above actually changes the estimated values of states and actions. Is this necessary? Suppose the bonus $\kappa\sqrt{\tau}$ was used not in updates, but solely in action selection. That is, suppose the action selected was always that for which $Q(S_t, a) + \kappa\sqrt{\tau(S_t, a)}$ was maximal. Carry out a gridworld experiment that tests and illustrates the strengths and weaknesses of this alternate approach. □



Exercise 8.5 How might the tabular Dyna-Q algorithm shown on page 164 be modified to handle stochastic environments? How might this modification perform poorly on changing environments such as considered in this section? How could the algorithm be modified to handle stochastic environments *and* changing environments? □

Extensions of Dyna-Q to stochastic environments is straightforward. The model needs to maintain counts of the number of times each state–action pair has been experienced and what the next states and rewards were. Naturally then, the planning is not done with a sample update as in the original version but as an expected update. Expected updates taking into account all possible next states, rewards and their relative frequencies of occurrence observed during the environment interaction.

Keeping sample averages state-action transitions doesn't perform well on non-stationary environments because all encountered samples have the same weight irrespectively when they were observed. A fixed-step alpha update with a reasonable step-size would give more weight to the recent values and allow for tracking of slowly changing non-stationary environments.

Exercise 8.6 The analysis above assumed that all of the b possible next states were equally likely to occur. Suppose instead that the distribution was highly skewed, that some of the b states were much more likely to occur than most. Would this strengthen or weaken the case for sample updates over expected updates? Support your answer. □

Highly skewed action probabilities would strengthen the case for sample updates. High-probability transitions are sampled frequently and the sampling update would receive the vast majority of the accuracy with a small amount of computation and samples.

The remaining low-probability situations don't help much in reducing RMS error but come at a fixed computational cost. Cost per additional branching state is the same irrespectively of it's weight in the overall sum.

Exercise 8.7 Some of the graphs in Figure 8.8 seem to be scalloped in their early portions, particularly the upper graph for $b = 1$ and the uniform distribution. Why do you think this is? What aspects of the data shown support your hypothesis? □

```
Scalloping is caused by zero initialization of the action value function Q and the
sweeping updates. Sequential jumps from a state to the next is very unlikely to
encounter an actual transition in sequence. In expectation half of the states during
the first sweep haven't received an updated yet. Event when a state receives an updates
it's negative in half of the case but since the other action may have not yet been non-
trivially updated, it's therefore possible that the negative rewards are temporarily
overlooked by the better promise of zero initialization.
```

Exercise 8.8 (programming) Replicate the experiment whose results are shown in the lower part of Figure 8.8, then try the same experiment but with $b = 3$. Discuss the meaning of your results. □