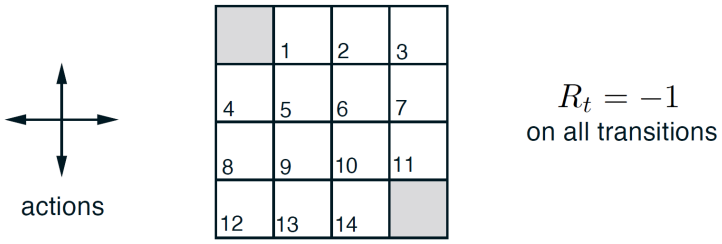
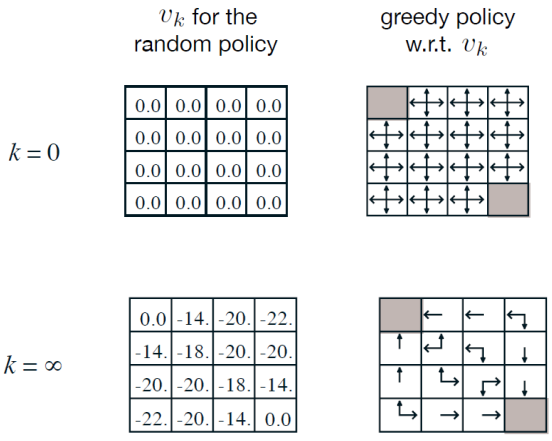
Exercises - Chapter 4

Victoria Nagorski

Exercise 4.1

In Example 4.1, if π is the equiprobable random policy, what is $q_\pi(11, down)$? What is $q_\pi(7, down)$?





From exercise 3.13:

$$q_\pi(s, a) = \sum_{s', r} \underbrace{p(s', r|s, a)}_{\text{0 or 1}}[r + \gamma v_\pi(s')]$$

Reward for every action is always -1 unless in a terminal state. State 11 has one negative reward before it reaches terminal state if it goes down. All 7's actions will bring a negative reward of -1.

State 11, set $\gamma = 1$:

$$q_\pi(11, down) = \sum_{s'} \underbrace{p(termination, -1|11, down)}_{1}[-1 + \underbrace{v_\pi(termination)}_{0}]$$

$$q_\pi(11, down) = -1$$

State 7, set $\gamma = 1$:

$$q_\pi(7, down) = \sum_{s'} \underbrace{p(11, -1|7, down)}_{1}[-1 + \underbrace{v_\pi(termination)}_{@k=\infty,\ v_\pi(11)=-14}$$

$$q_\pi(7, down) = -15,\ @k = \infty$$

$$q_\pi(7, down) = -1,\ @k = 0$$

Exercise 4.2

In Example 4.1, suppose a new state 15 is added to the gridworld just below state 13, and its actions, left, up,right, and down, take the agent to states 12, 13, 14, and 15, respectively. Assume that the transitions from the original states are unchanged. What, then, is $v_pi(15)$ for the equiprobably random policy? Now suppose the dynamics of state 13 are also changed, such that action down from state 13 takes the agent to the new state 15. What is $v_pi(15)$ for the equiprobable random policy in this case>

$\gamma = 1$ and $s = 15$:

$$a(left) \rightarrow 12$$

$$a(up) \rightarrow 13$$

$$a(right) \rightarrow 14$$

$$a(down) \rightarrow 15$$

$$v_\pi(s) = \underbrace{\sum_{a}(a|s)}_{0.25 \text{ each}} \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

Sum over the different values by the equiprobable policy. Since $v_\pi(15)$ does not have a value yet, I will use -20 since has the same outcomes as 13.

$$v_\pi(15) = \sum_{s'}(0.25)[-1 + v_\pi(s')]$$

$$v_\pi(15) = \underbrace{(0.25)[-1 - 22]}_{left} + \underbrace{(0.25)[-1 - 20]}_{up} + \underbrace{(0.25)[-1 - 14]}_{right} + \underbrace{(0.25)[-1 - 20]}_{down}$$

$$v_\pi(15) = -20$$

The state value would remain the same since $v_\pi(15)$ has the same value as the $v_\pi(13)$ state. With the previous physics, a down motion would have caused the agent to stay in the 13^{th} state.

Exercise 4.3

What are the equations analogous to (4.3), (4.4), and (4.5) for the action-value function q_π and its successive approximation by a sequence of functions q_0, q_1, q_2,...?

$$q_{k+1}(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')]$$

Exercise 4.4

The policy iteration algorithm on page 80 has subtle bug in that it may never terminate if the policy continually switches between two or more policies that are equally good. This is ok for pedagogy, but not for actual use. Modify the pseudocode so that convergence is guaranteed.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

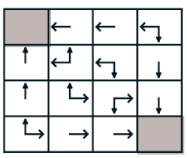
2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

If you look at figure 4.1 when $k = \infty$:

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



A set of optimal policies may converge much quicker than the v_k values. However, because there are multiple sets of policies that lead to the same v_k values, the current psuedocode may not end because it keeps switching between the optimal policies.

The psuedocode may be modified so that V before Policy Improvement is compared to V after Policy Improvement. You could also use the following condition statement:

$$q_\pi(s, \pi'(s)) \geq v_\pi(s), s \in \mathcal{S}$$

.

Exercise 4.5

How would policy iteration be defined for action values? Give a complete algorithm for computing $q_\star$, analogous to that on page 80 for computing $v_\star$. Please pay special attention to this exercise, because the ideas will be used throughout the rest of the book.

Algorithm 1 Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_\star$

1. Initialization
   $Q(s,a) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:

   $\Delta \leftarrow 0$

   Loop for each $s \in \mathcal{S}$:

   $q \leftarrow Q(s,a)$
   $Q(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$
   $\Delta \leftarrow max(\Delta, |q - Q(s)|)$

   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   $policy - stable \leftarrow true$
   For each $s \in \mathcal{S}$:

   $old - action \leftarrow \pi(s)$

   $\pi(s) \leftarrow \underset{a}{\text{argmax}}\ q_\star(s,a)$
   If $old - action \neq \pi(s)$, then $policy - stable \leftarrow false$

If $policy - stable$, then stop and return $Q \approx q_\star$ and $\pi \approx \pi_\star$; else go to 2

Exercise 4.6

Suppose you are restricted to considering only policies that are $\varepsilon - soft$, meaning that the probability of selecting each action in each state, s, is at least $\varepsilon/|\mathcal{A}(s)|$. Describe qualitatively the changes that would be required in each of the steps 3, 2, and 1, in that order,of the policy iteration algorithm for $v_\star$ on page 80.

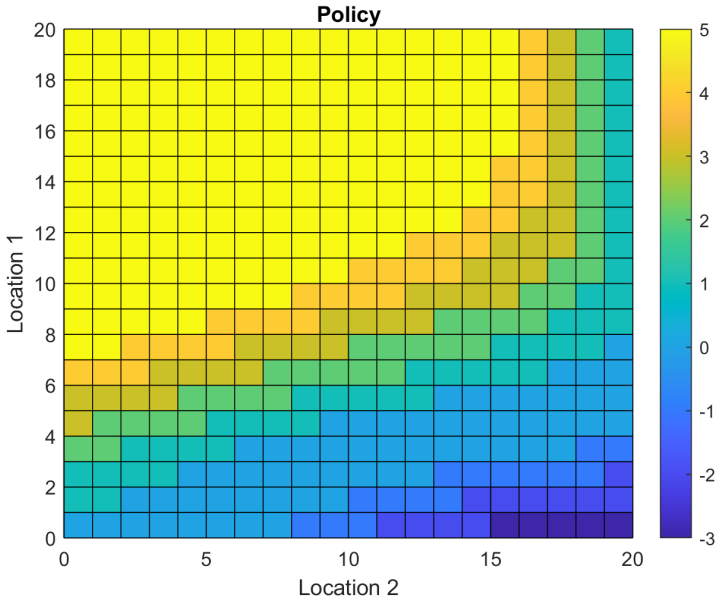
Step 3:
Each state would not have to carry a policy that would allow for a decision to made off of the probability of that action. Instead of updating a hard "do *action* at state *blank*" it now needs to update an array of numbers that represent the probability for each action in that state.

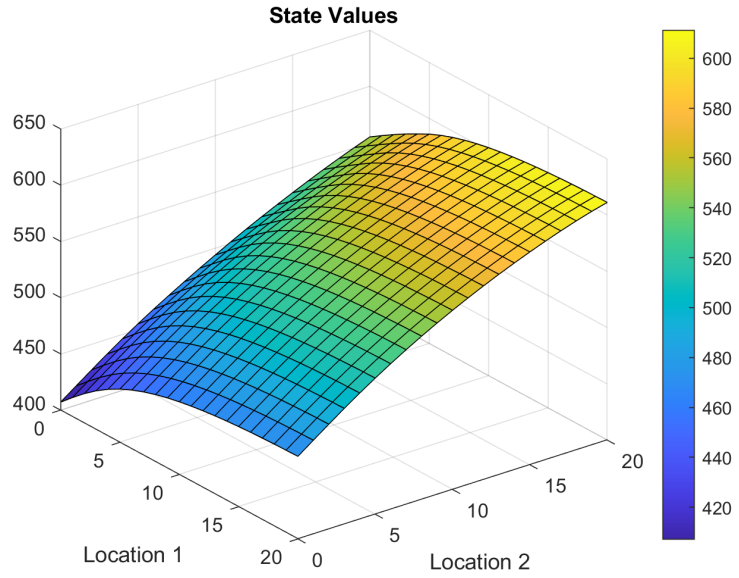
Step 2:
When choosing the action when solving for the new V_π, the algorithm would have to choose based of the probabilistic policy. In the current algorithm, the the policy is one dimensional for each state.

Step 1:
The policy would need to initialized to cover the state, but then have an extra dimension for the actions.

Exercise 4.7

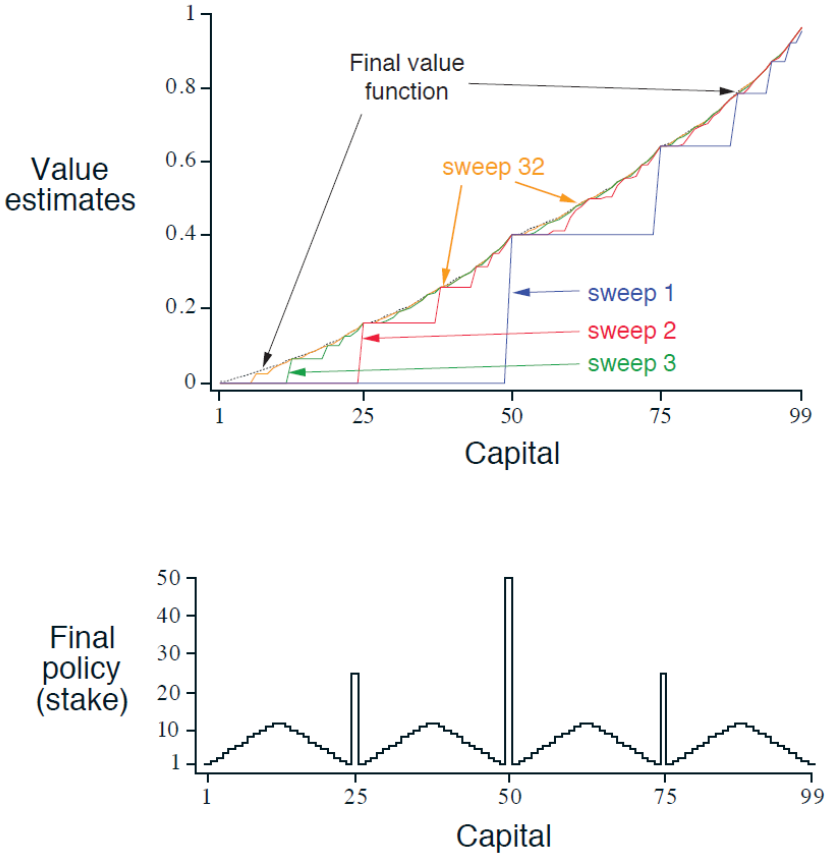
Write a program for policy iteration and re-solve Jack's car rental problem with the following changes. One of Jack's employees at the first location rides a bus home each night and lives near the second location. She is happy to shuttle one car to the second location for a fee. Each additional car still costs $\$2$, as do all cars moved in the other direction. In addition, Jack has limited parking space at each location. If more than 10 cars are kept overnight at a location (after any moving of cars), then an additional cost of $\$4$ must be incurred to use a second parking lot (independent of how many cars are kept there). These sort of nonlinearities are arbitrary dynamics often occur in real problems and cannot easily be handled by optimization methods other than dynamic programming. To check your program, first replicate the results given for the original problem.





Exercise 4.8

Why does the optimal policy for the gambler's problem have such a curious form? In particular, for capital of 50 policy for the gambler's problem have such a curious form? In particular, for capital of 50 it bets it all on one flip, but for capital of 51 it does not. Why is this a good policy?



When the agent is a 50 capital, there is a sudden spike. At 25 and 75, there are notable (but smaller) spikes on either side of the 50 capital mark.

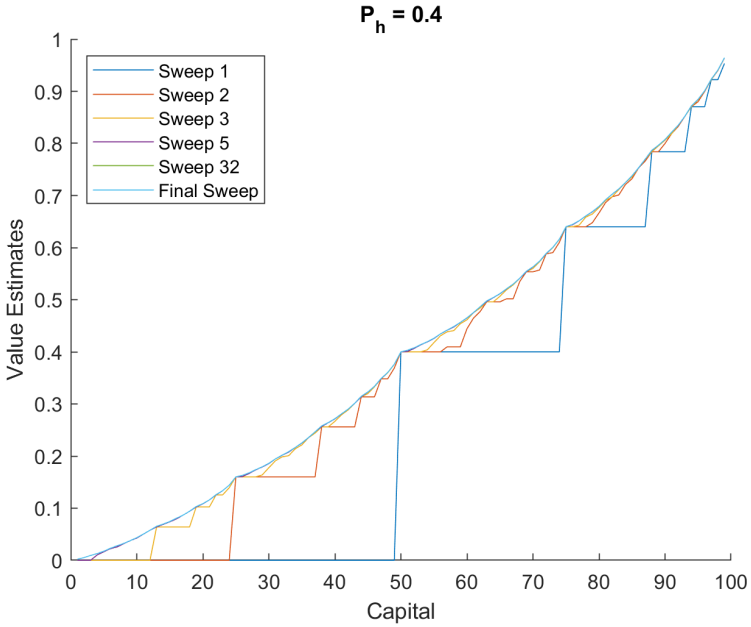
The reason the agent would do this is because it only receives its reward once it accumulates $100. There is no extra reward for $102, so it saves the extra dollar. This would allow the agent to stay in the game if the coin were to flip tails (60% chance).

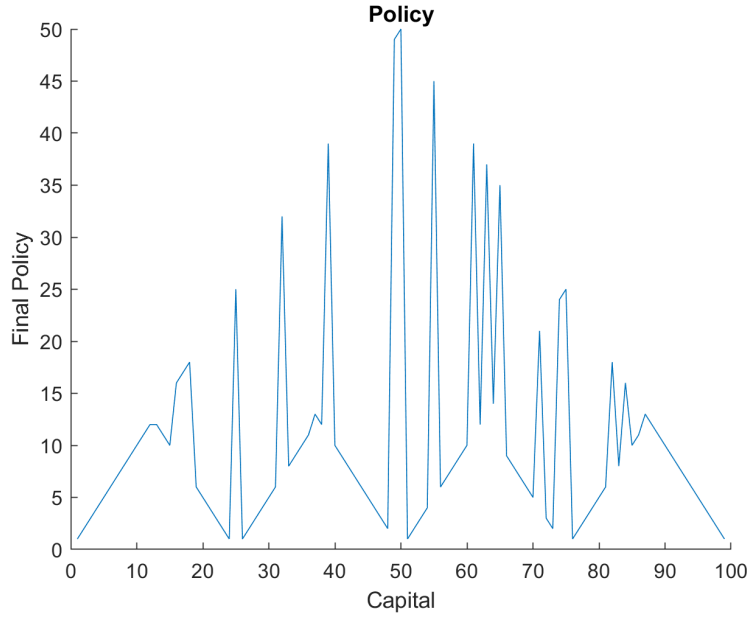
The 25 and 50 mark are sort of the optimal time to make a risky gamble. 50 is the optimal place. Betting it all at 25 would get the agent there 40% of the time. At 75, it only need to put down 25. If it looses, it gets put back to 50- where it will bet it all again.

Exercise 4.9

Implement value iteration for the gambler's problem and solve it for $p_h = 0.25$ *and* $p_h = 0.55$. *In programming, you may find it convenient to introduce two dummy states corresponding to termination with capital of 0 and 100, giving them values of 0 and 1 respectively. Show your results graphically, as in Figure 4.3. Are your results stables at* $\theta \to 0$*?*

Before doing Exercise 4.9, I first verified my code against the example problem:

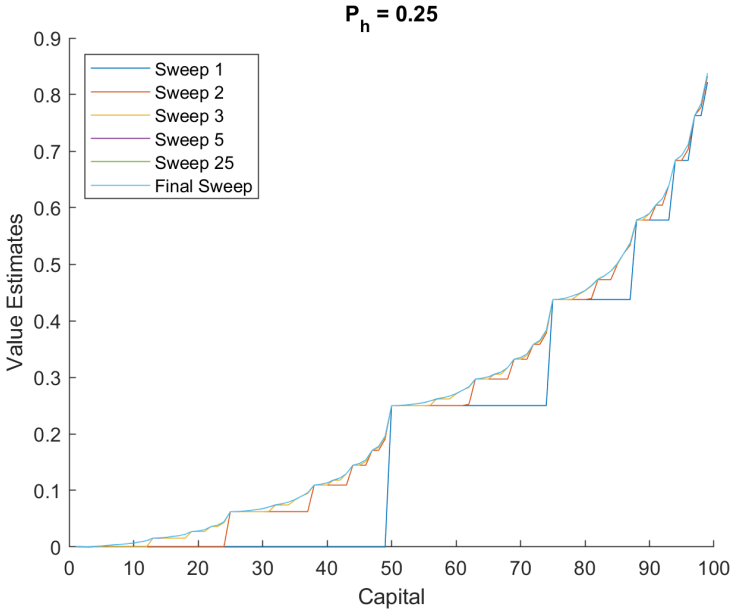


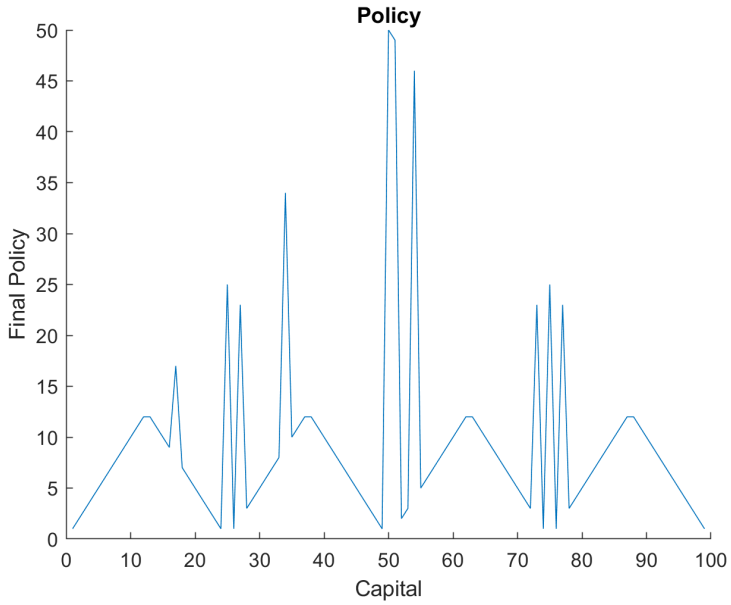


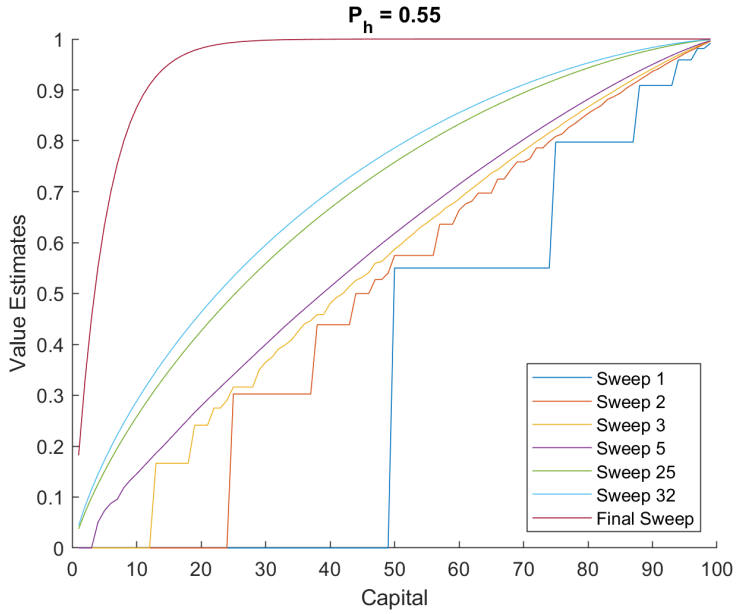
As you can see, my figures do not exactly align with the the authors figures. My graph for the state values converge much faster, and my policy graph has more noise than the authors'. However, I believe the author might have used a slightly different algorithm than what was given in the psuedocode- which would account for this difference.

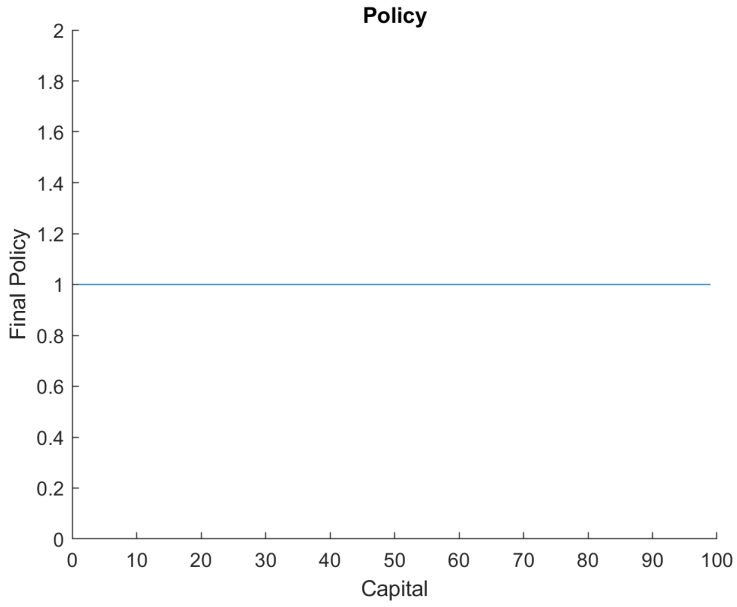
If you think it through: The convergence I see on value estimate graph makes sense if you look at how the algorithm on psuedocode converges to the "correct" value. For example, if you currently have $25, you can gamble it all and theoretically see the $50 value. Since there is probability of seeing the $50 state value, the $25 state value will update based off of the $50 state value. Then once you have a state value other than 0 at $25, the halfway mark between $0 and $25 can update with a state value.

This pattern should consistent with every sweep. In my graphs, this pattern is visible and very close to visibly converging around sweep 5. By the 32nd sweep, you cannot see difference between the 32nd sweep and the final sweep. Looking at the author's state value graph, we see that his graphs initially follow this pattern, but then stop at the 32nd sweep. If the author was using the same algorithm, it would not make sense for there to still be 0s on the first few state values for the 32nd sweep.









Exercise 4.10

What is the analog of the value iteration update (4.10) for action values, $q_{k+1}(s,a)$?

Equation 4.2:

$$q_\star = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_\star(s',a')]$$

Therefore:

$$q_{k+1} = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_k(s',a')]$$