Exercise 0
Gregory Attra
02.20.2022
CS 7180 - Prof. Amato

1. (a) $V^*(s) = \max_a\{q^*(s,a)\}$

   (b) $q^*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'}\{q^*(s',a')\}$

   (c) $\pi^*(s) = \arg\max\{q^*(s,a)\}$

   (d) $\pi^*(s) = \arg\max_a\{\sum_{s',r} p(s',r|s,a)[R(s,a) + \gamma V^\pi(s')\}$

   (e) i. $V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[R(s,a) + \gamma V^\pi(s')]$
      ii. $q^\pi(s,a) = \sum_{s'} p(s'|s,a)[R(s,a) + \gamma \max_{a'}\{q^\pi(s',a')\}$
      iii. $V^*(s) = \sum_{s'} p(s'|s,\pi^*(s))[R(s,\pi^*(s)) + \gamma V^*(s')]$
      iv. $q^*(s,a) = \sum_{s'} p(s'|s,a)[R(s,a) + \gamma \max_{a'}\{q^*(s',a')\}$

2. (a) The bug is in the policy improvement step (3). If the value of the new policy action is different than the old policy action, but the values for each action are equal, the policy will claim to be unstable despite having two equally good policies. The fix would be to change the conditional to be:

```
**if** $\pi(s) \neq old_action\mathbf{AND}\sum_{s',r} p(s',r|s,\pi'(s))[r+\gamma V(s')] > \sum_{s',r} p(s',r|s,\pi(a))[r+\gamma V(s')]$ **then**
   policy-stable = False
**end if**
```

   (b) No similar bug exists. We iterate until our value function converges (which is guaranteed). Then we set the action for a given state greedily using the value function to pick the action with the highest value. If this action is different from our existing policy action, as long as it has the same value as the existing policy action, the algorithm still terminates.

3. (a) i. Initialize:

```
initialize $q^\pi(s,a) \leftarrow arbitrarily\ \forall s \in S, a \in A$
initialize $\pi(s) \leftarrow arbitrarily\ \forall s \in S$
```

   ii. Evaluate:

```
$\Delta = 0$
while $\Delta > \theta$ do
   for $s \in S$ do
      for $a \in A$ do
         $q = q^\pi(s,a)$
         $q^\pi(s,a) = \pi(a,|s) \sum_{s',r} p(s',r|s,a)[r+\gamma \max_{a'}\{q^\pi(s',a')]$
         $\Delta \leftarrow \max(\Delta|q - q^\pi(s,a))$
      end for
   end for
end while
```

iii. Improve:

```
for $s \in S$ do
    $\pi(s) = \arg\max_a \{q^\pi(s, a)\}$
end for
```

(b) $q^\pi_{k+1}(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} \{q^\pi_k(s', a')\}]$

4. (a) It is better to be in state x than y. Size z is the goal state, and while both x and y have the same probability of entering z, the penalty of remaining in y is twice as severe as that of x. The optimal policy will likely move to state x first, then attempt to move to state z, as it will incur a smaller expected negative reward than if it were to remain in y and attempt to enter z repeatedly.

   (b) See images below.

   (c) i. It takes longer for policy iteration to converge to the optimal policy as we must go further toward the limit for $V(x)$ to outweigh $V(y)$.

   ii. Discounting does help. If $\gamma \leftarrow 0$, evaluation converges after one step:
   $V_{xc} = 0.9(-1) = 0.9$
   $V_{xb} = 0.8(-2) + 0.2(-1) = -1.8$
   $V_{yc} = 0.9(-2) = -1.8$
   $V_{yb} = 0.8(-1) + 0.2(-2) = -1.2$

   iii. It is not dependant on discounting. I showed in $4.a$ that we converge when $\gamma = 1$, and I showed in the above example that we converge when $\gamma = 0$. If we converge when $\gamma = 0 \& \gamma = 1$, then we will converge when $\gamma \in [0, 1]$

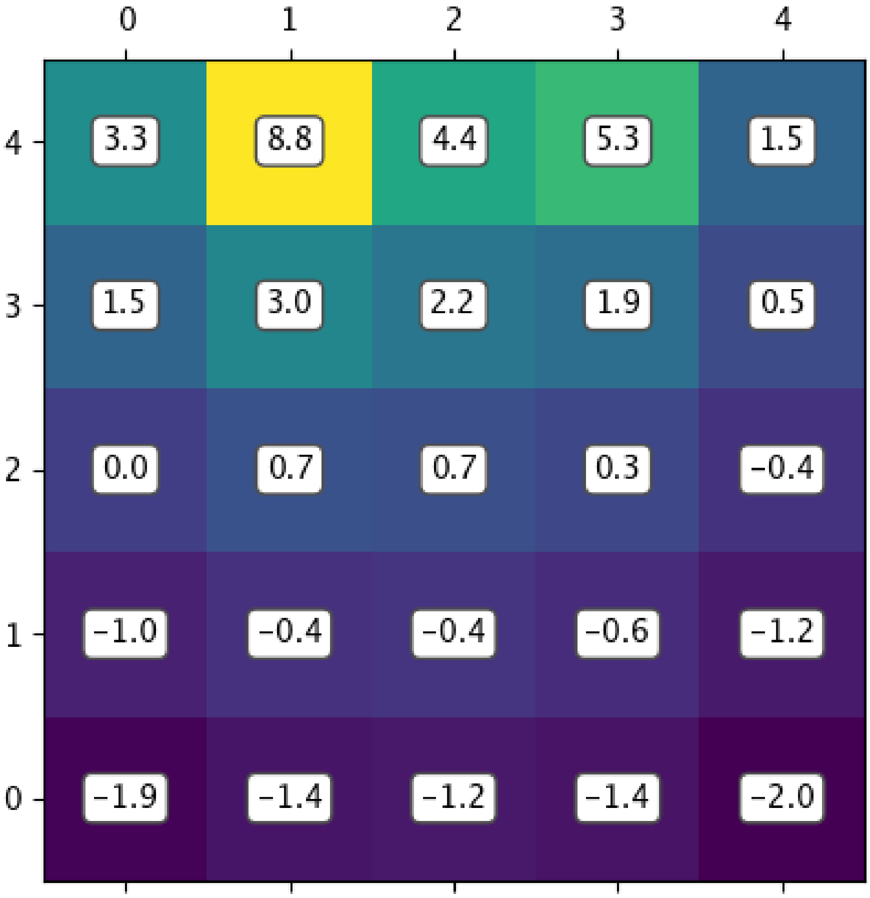
5. Plots are on the pages below.
   Setup:

   - Unzip the code files
   - ‘cd‘ into the ‘code‘ directory
   - if on Ubuntu, you may need to run ‘./install-ubuntu-deps‘
   - ‘source ./init‘ the environment

   (a) To run policy evaluation, run ‘python src/grid_world/run_policy_evaluation.py‘

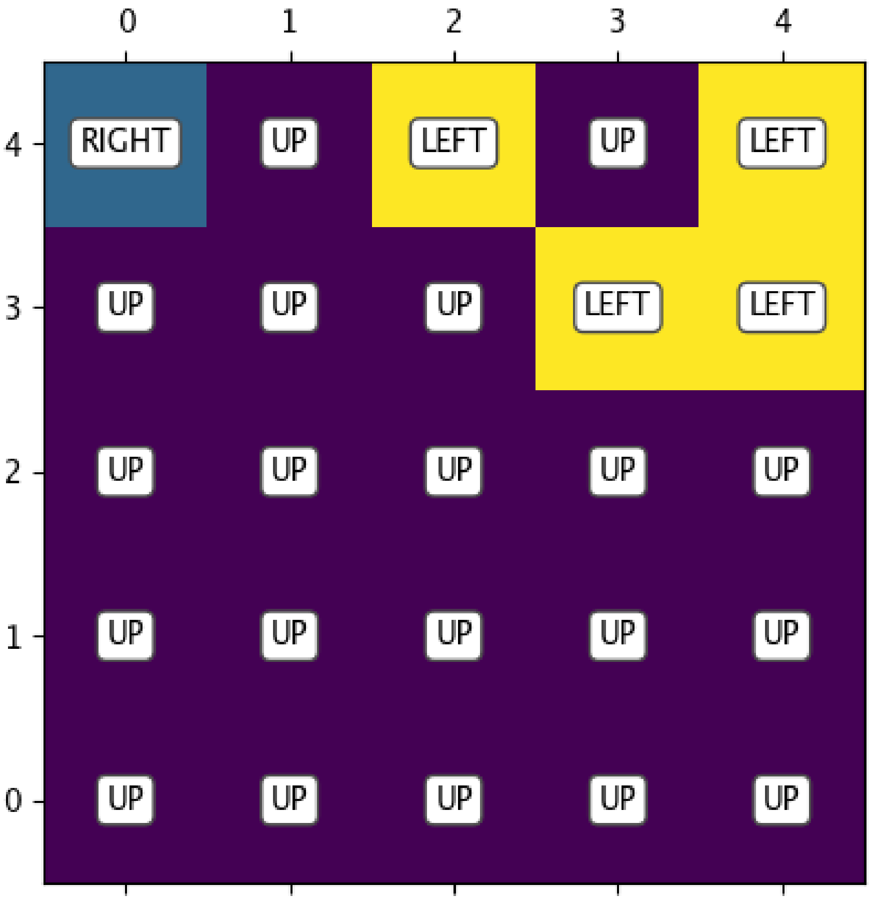
   (b) To run policy iteration run ‘python src/grid_world/run_policy_iteration.py‘

   (c) To run value iteration, run ‘python src/grid_world/run_value_iteration.py‘

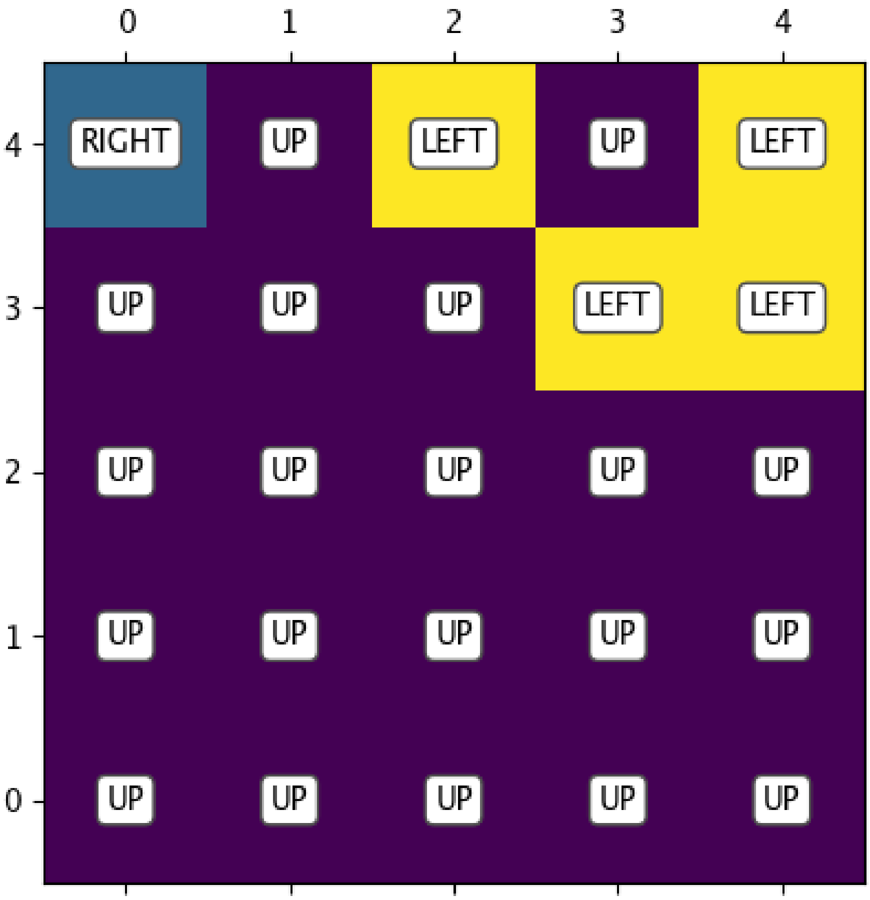


	0	1	2	3	4
4	3.3	8.8	4.4	5.3	1.5
3	1.5	3.0	2.2	1.9	0.5
2	0.0	0.7	0.7	0.3	−0.4
1	−1.0	−0.4	−0.4	−0.6	−1.2
0	−1.9	−1.4	−1.2	−1.4	−2.0

Policy Evaluation



Policy Iteration



	0	1	2	3	4
4	RIGHT	UP	LEFT	UP	LEFT
3	UP	UP	UP	LEFT	LEFT
2	UP	UP	UP	UP	UP
1	UP	UP	UP	UP	UP
0	UP	UP	UP	UP	UP

Value Iteration

6. Plots are on the pages below.

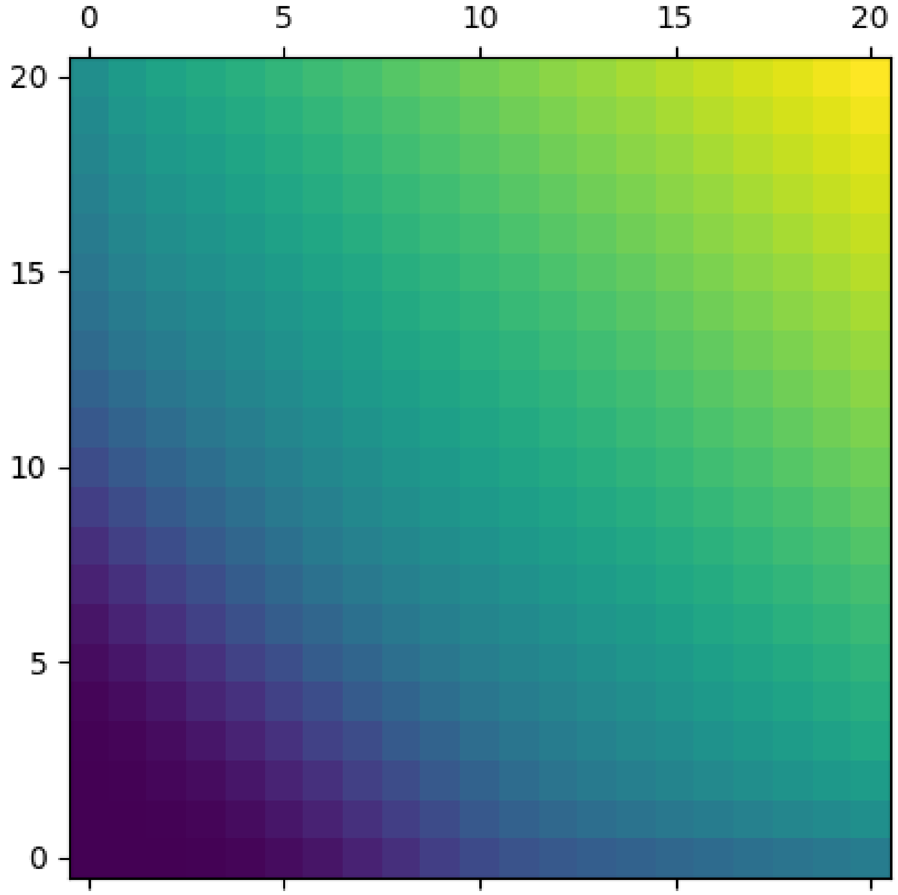
   Setup:

   - Unzip the code files
   - ‘cd‘ into the ‘code‘ directory
   - if on Ubuntu, you may need to run ‘./install-ubuntu-deps‘
   - ‘source ./init‘ the environment

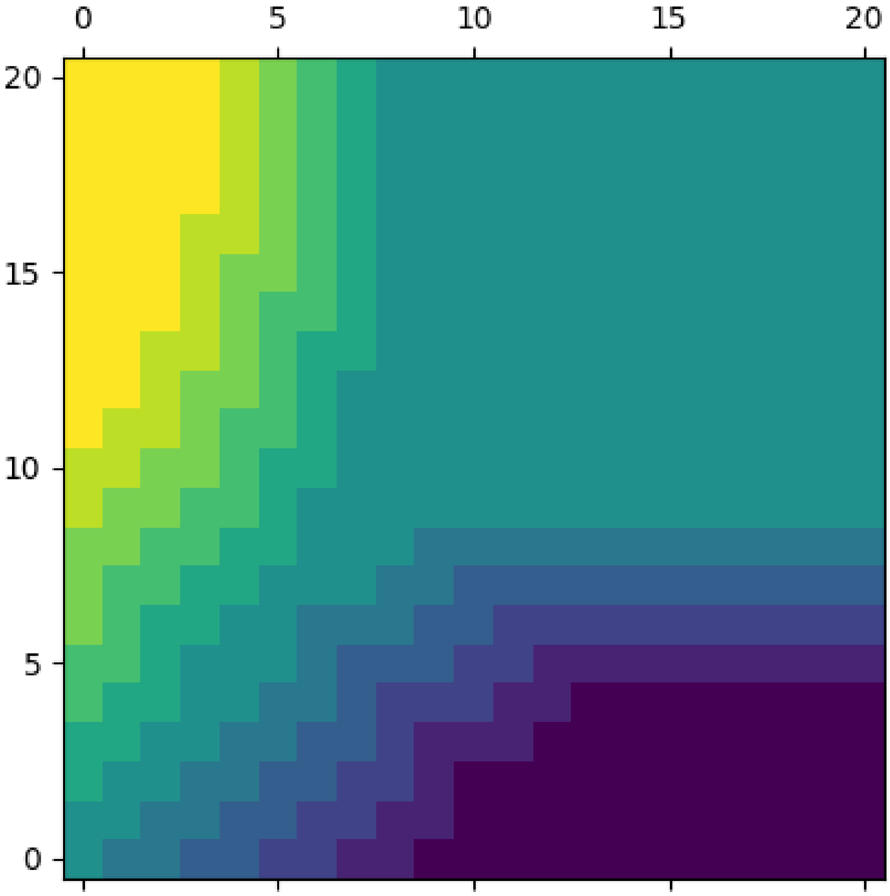
   (a) I was unable to get my policy to converge to the optimal policy. I am confident this is due to my implementation of the domain dynamics and not policy iteration as I was able to come close to the optimal policy. And my value function plot looks correct.

   (b) To run policy iteration on Jack’s rental car problem:, run ‘python src/rental_car/run_normal.py‘

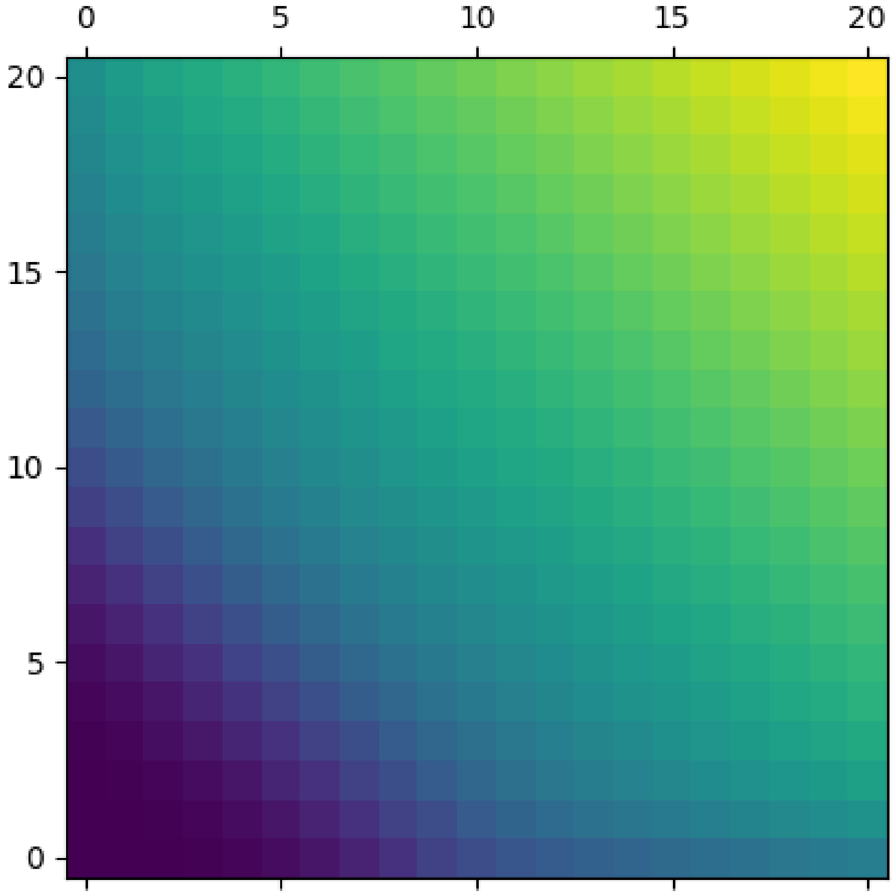
   (c) To run policy iteration on Jack’s rental car problem, with the modifications to the problem, run ‘python src/rental_car/run_modified.py‘



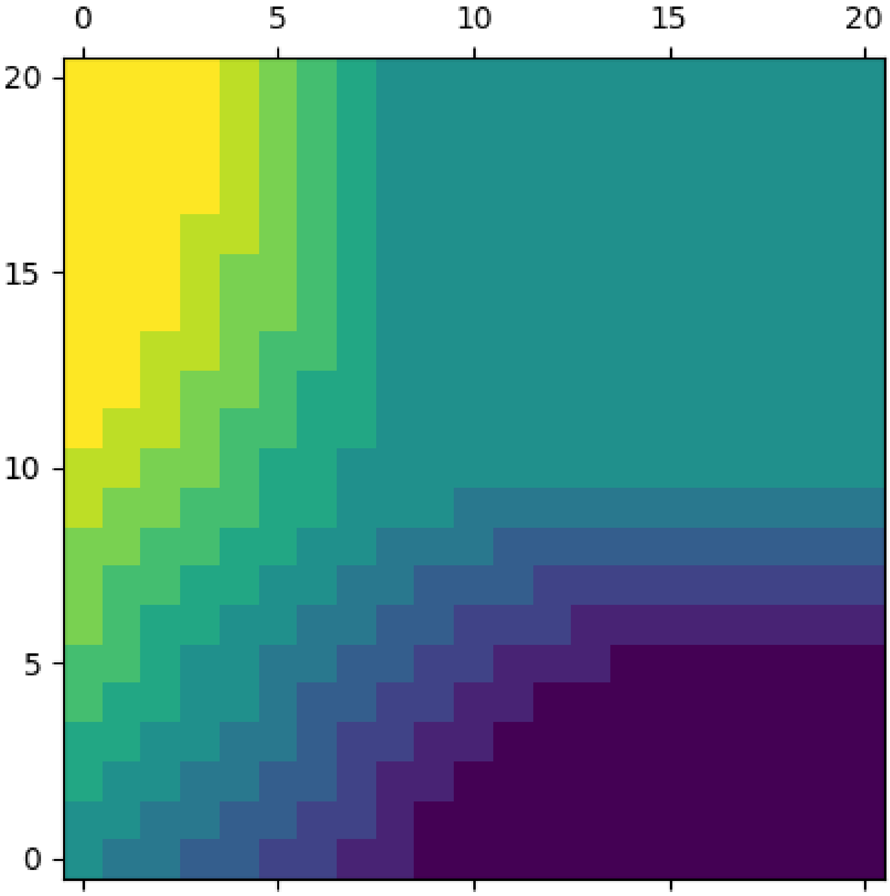
Jack’s Rental Car - Value



Jack’s Rental Car - Policy

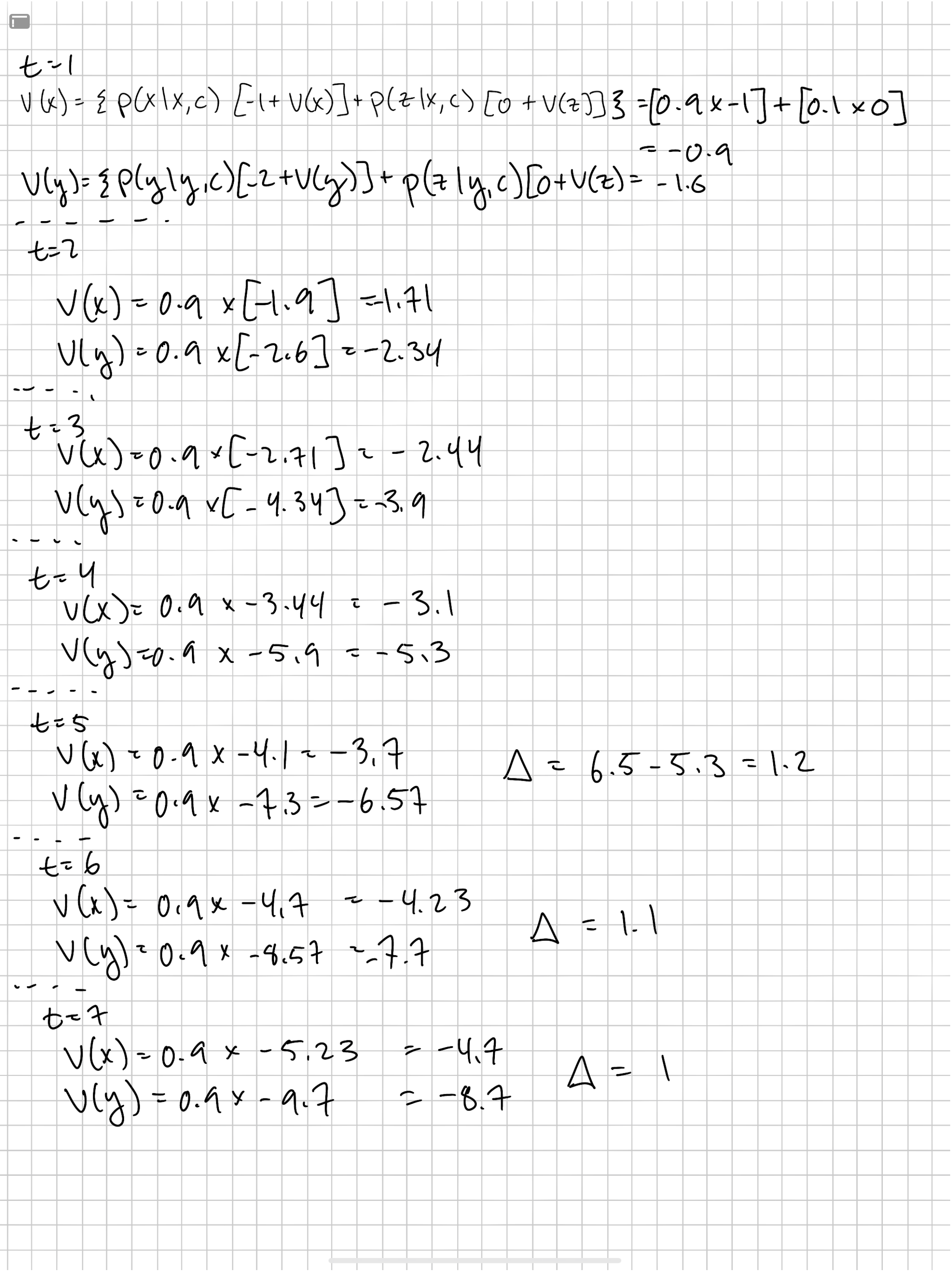


Modified Jack's Rental Car - Value



Modified Jack's Rental Car - Policy

Question 4.a: (images below)



$t=1$

$V(x) = \{ p(x|x,c)\,[-1+V(x)] + p(z|x,c)\,[0+V(z)] \} = [0.9 \times -1] + [0.1 \times 0]$

$= -0.9$

$V(y) = \{ p(y|y,c)[-2+V(y)] + p(z|y,c)[0+V(z) = -1.6$

- - - - - -

$t=2$

$V(x) = 0.9 \times [-1.9] = 1.71$

$V(y) = 0.9 \times [-2.6] = -2.34$

- - - - -

$t=3$

$V(x) = 0.9 \times [-2.71] = -2.44$

$V(y) = 0.9 \times [-4.34] = -3.9$

- - - -

$t=4$

$V(x) = 0.9 \times -3.44 = -3.1$

$V(y) = 0.9 \times -5.9 = -5.3$

- - - - -

$t=5$

$V(x) = 0.9 \times -4.1 = -3.7$ $\quad \Delta = 6.5 - 5.3 = 1.2$

$V(y) = 0.9 \times -7.3 = -6.57$

- - - -

$t=6$

$V(x) = 0.9 \times -4.7 = -4.23$ $\quad \Delta = 1.1$

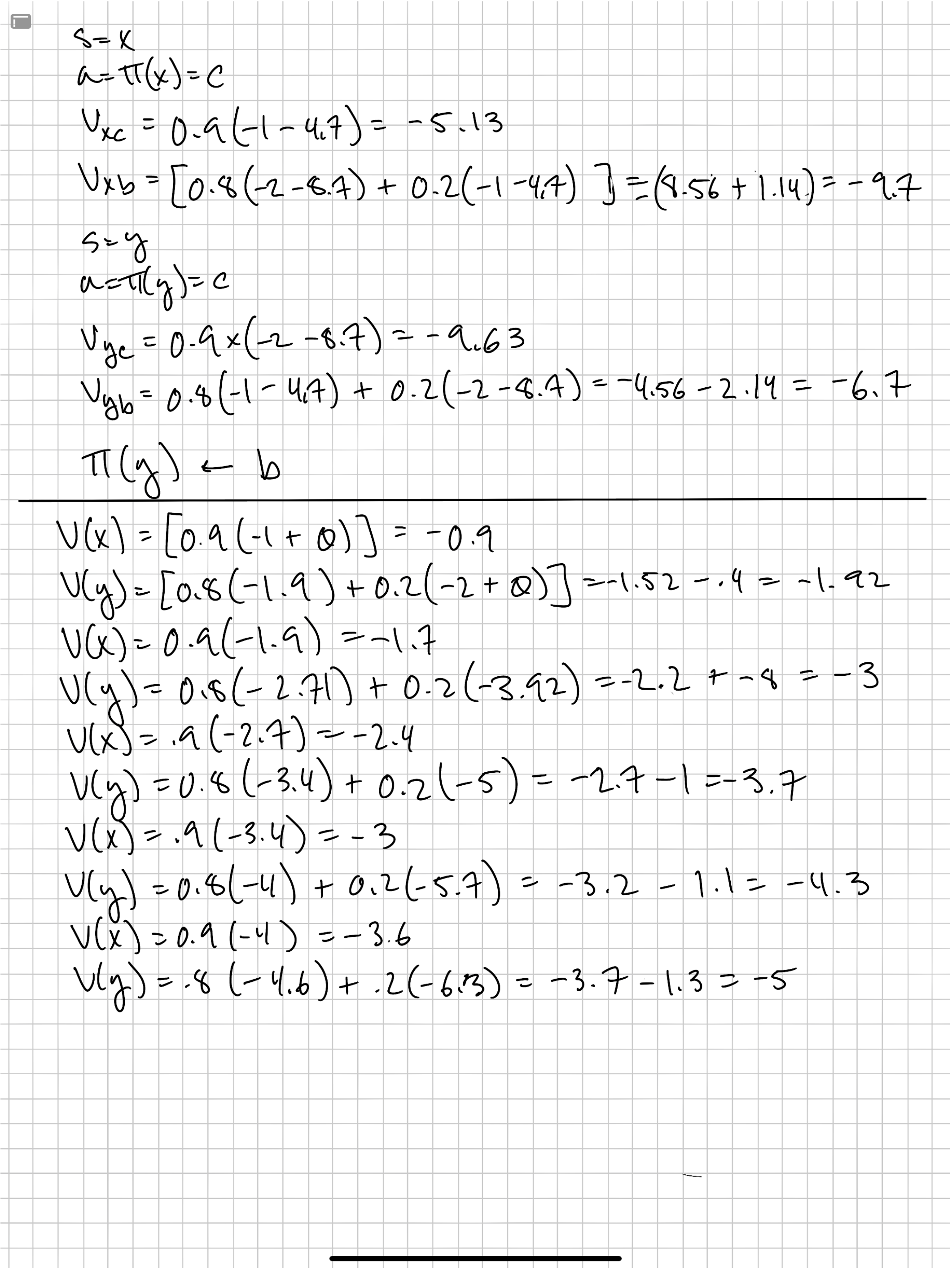
$V(y) = 0.9 \times -8.57 = -7.7$

- - - -

$t=7$

$V(x) = 0.9 \times -5.23 = -4.7$ $\quad \Delta = 1$

$V(y) = 0.9 \times -9.7 = -8.7$



$s = x$

$a = \pi(x) = c$

$V_{xc} = 0.9(-1 - 4.7) = -5.13$

$V_{xb} = [0.8(-2 - 8.7) + 0.2(-1 - 4.7)] = -(8.56 + 1.14) = -9.7$

$s = y$

$a = \pi(y) = c$

$V_{yc} = 0.9 \times (-2 - 8.7) = -9.63$

$V_{yb} = 0.8(-1 - 4.7) + 0.2(-2 - 8.7) = -4.56 - 2.14 = -6.7$

$\pi(y) \leftarrow b$

$V(x) = [0.9(-1 + 0)] = -0.9$

$V(y) = [0.8(-1.9) + 0.2(-2 + 0)] = -1.52 - .4 = -1.92$

$V(x) = 0.9(-1.9) = -1.7$

$V(y) = 0.8(-2.71) + 0.2(-3.92) = -2.2 + -.8 = -3$

$V(x) = .9(-2.7) = -2.4$

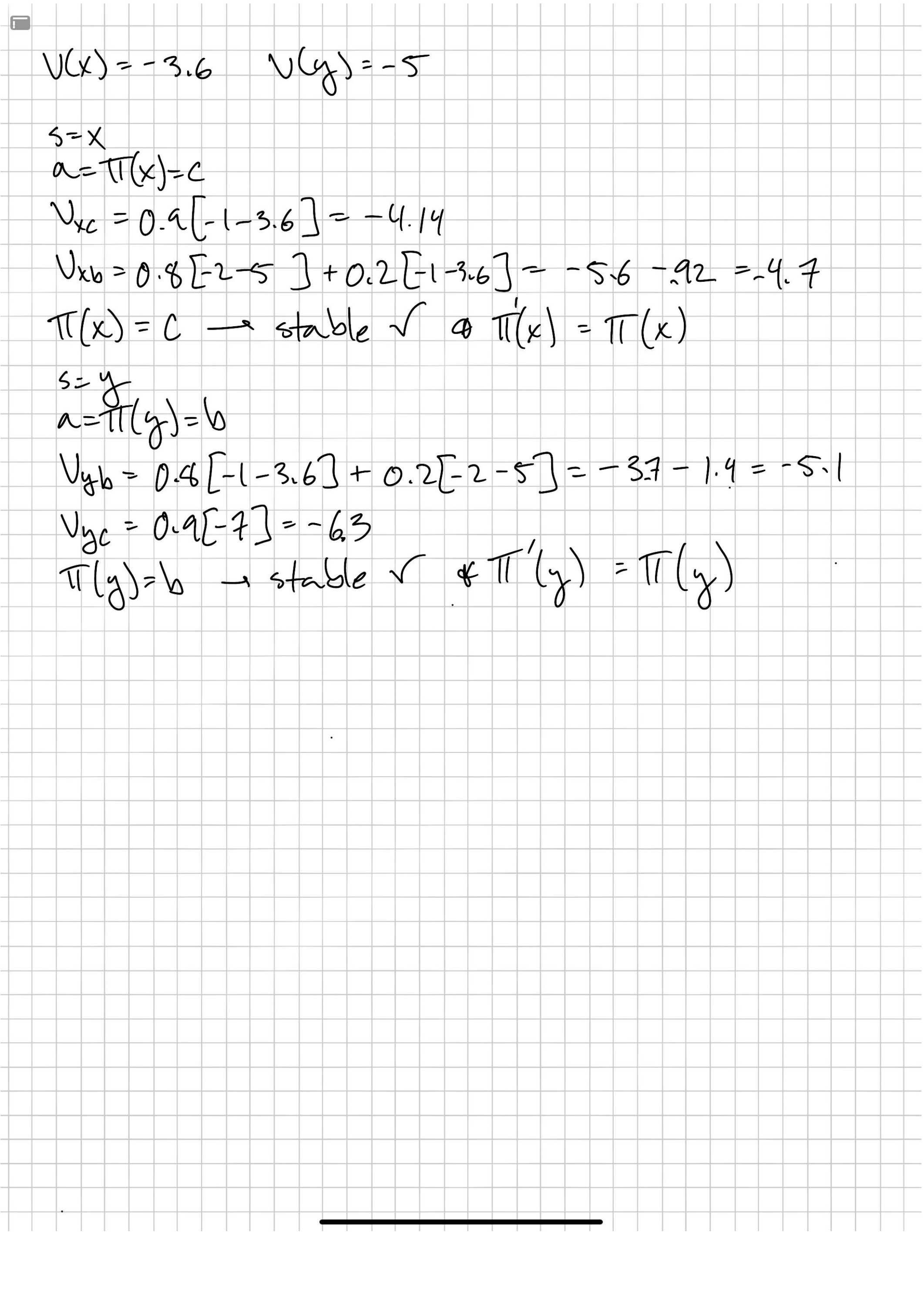
$V(y) = 0.8(-3.4) + 0.2(-5) = -2.7 - 1 = -3.7$

$V(x) = .9(-3.4) = -3$

$V(y) = 0.8(-4) + 0.2(-5.7) = -3.2 - 1.1 = -4.3$

$V(x) = 0.9(-4) = -3.6$

$V(y) = .8(-4.6) + .2(-6.3) = -3.7 - 1.3 = -5$



$V(x) = -3.6 \quad V(y) = -5$

$s = x$

$a = \pi(x) = c$

$V_{xc} = 0.9[-1-3.6] = -4.14$

$V_{xb} = 0.8[-2-5] + 0.2[-1-3.6] = -5.6 - .92 = -4.7$

$\pi(x) = c \longrightarrow$ stable $\checkmark$ & $\pi'(x) = \pi(x)$

$s = y$

$a = \pi(y) = b$

$V_{yb} = 0.8[-1-3.6] + 0.2[-2-5] = -3.7 - 1.4 = -5.1$

$V_{yc} = 0.9[-7] = -6.3$

$\pi(y) = b \rightarrow$ stable $\checkmark$ & $\pi'(y) = \pi(y)$