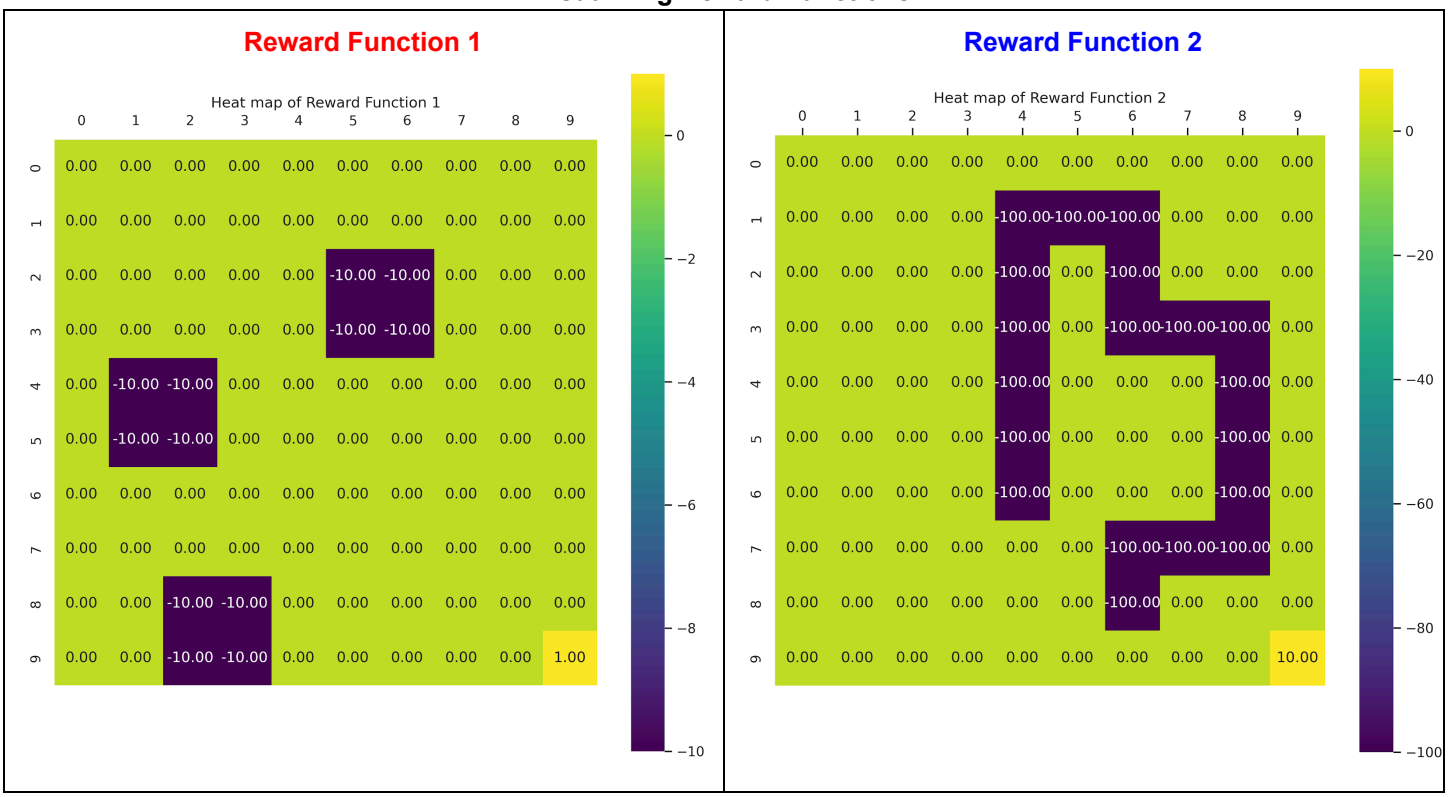
Project 3
Reinforcement Learning and Inverse Reinforcement Learning

Part 1: Reinforcement Learning (RL)

Visualizing Reward Functions

1) We are asked to plot a heat map of **Reward Function 1** and **Reward Function 2**. My plots are presented below in **Fig. 1**:

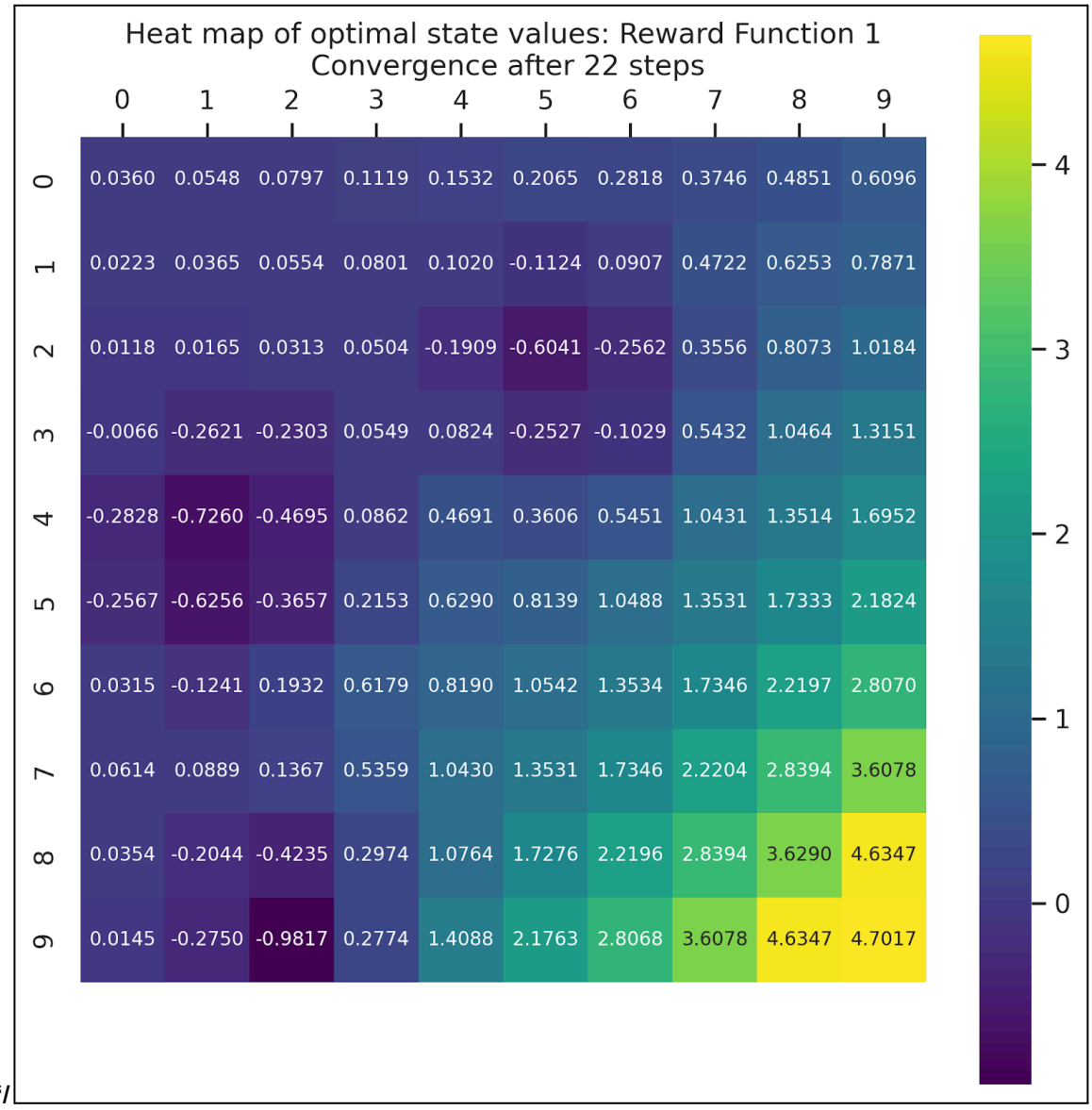
Fig. 1
Visualizing Reward Functions



Analyzing Optimal State-Values resulting from RL

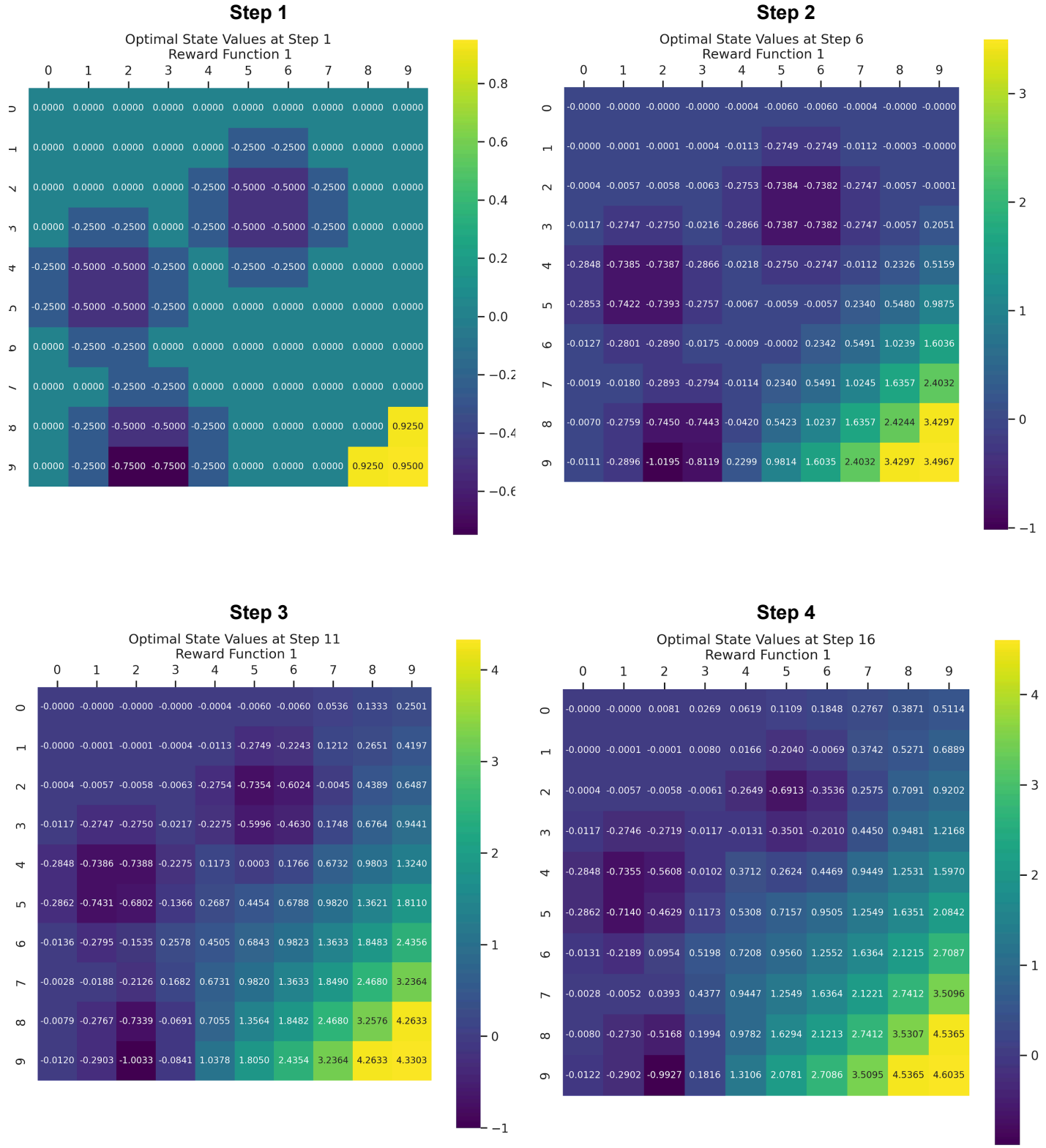
2) My optimal state-values converged after 22 steps **(N=22)** when using **Reward Function 1** and with an epsilon value of 0.01. We are asked to plot the optimal state value matrix after convergence which I have done below in **Fig. 2**

Fig. 2
Optimal State-Values: Reward Function 1
Convergence After 22 Steps

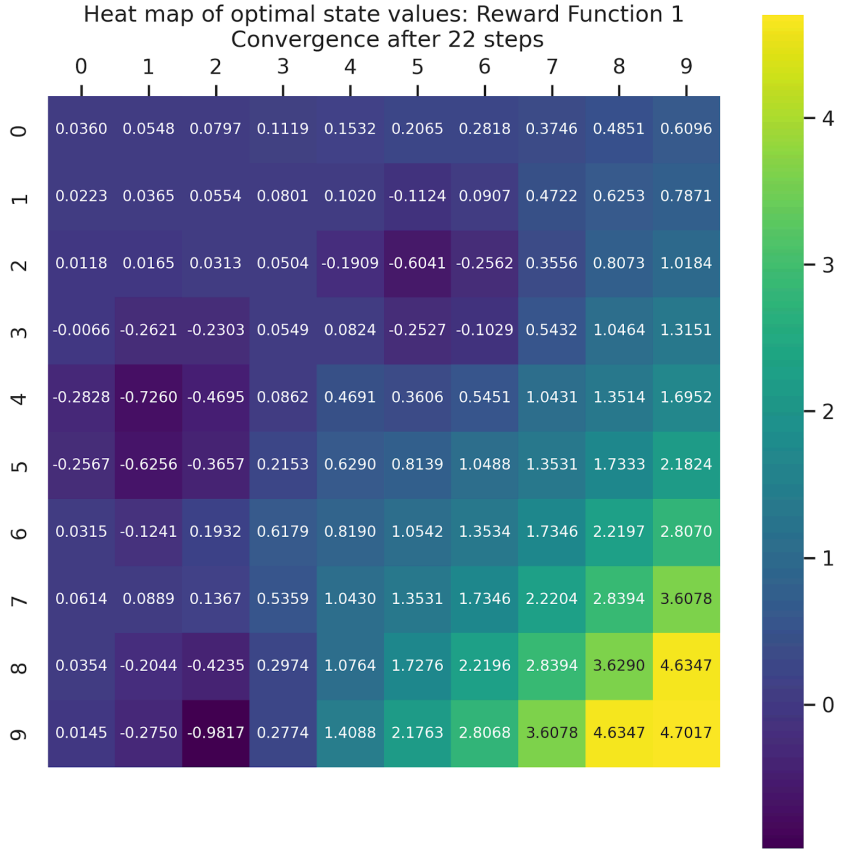


Next, we were asked to plot snapshots of the state value matrix in five different steps linearly distributed from 1 to N where N represents the step value where the state value matrix converged. My snapshots are presented below in Fig. 3

Fig. 3
Snapshots of state-values in five different steps
linearly distributed from 1-N (step value of convergence)



Step 5 - Convergence



Observations

- After the first step, most states have a value of 0. Only those states adjacent to a state with a reward other than 0 have values other than 0. The states adjacent to states with a negative reward value have negative values, while those adjacent to the states with a positive reward value have positive values.
- By the sixth step, all states have values other than 0.
- As steps increase, a pattern emerges and then grows around state 99 (the goal state), where states closer to this state are brighter. The three dark four-state blocks created by the states with a negative reward value fade as the steps increase, but by the 22 steps, when the optimal value converges, the pattern is still visible.

3) We are asked to plot a **heat map of the optimal state values**. My heat map is presented as **Step 5 above in Fig. 3** and also in Fig. 2

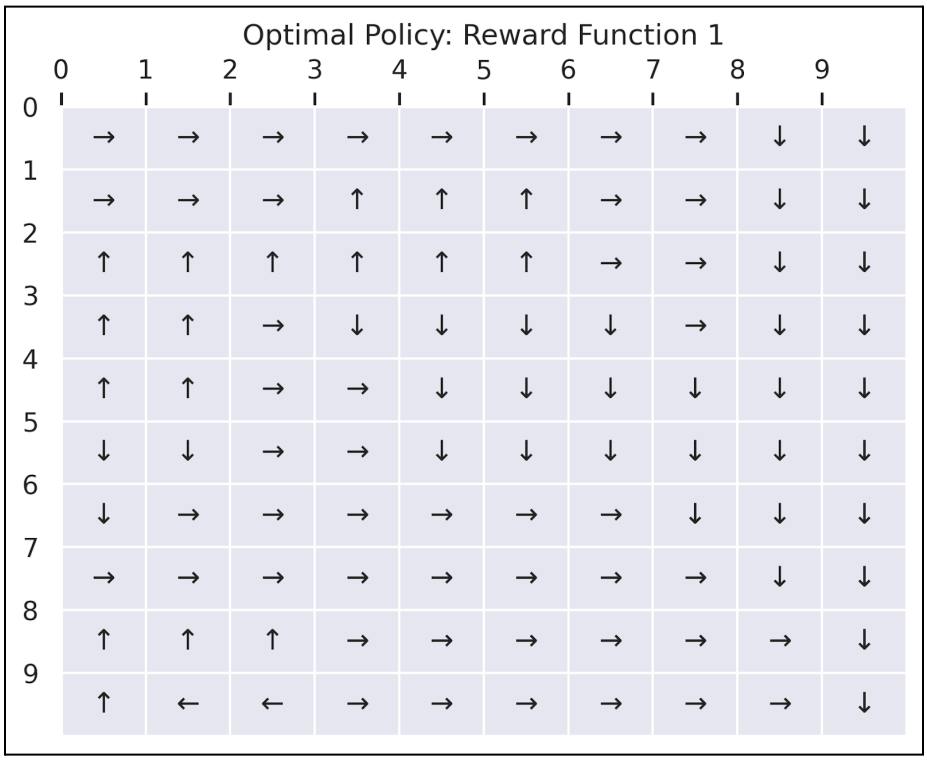
4) We can make several observations from the **heat map** presented above as **Step 5 in Fig. 3.**

- The state with the brightest color - state 99 - is also the state with the highest reward (1). Furthermore, state 99 is the only state with a positive reward. The closer a state is to state 99, the brighter it is, representing a higher optimal value.
- The state with the darkest color, state 92, is one of the states with the lowest reward value (-10). Furthermore, the pattern of the reward function, which has three 4-state blocks with reward values of (-10), can almost be made out in the heat map above. With a few exceptions, the states with the lowest reward values are also the darkest.
- The above patterns suggest that the reward function could be extracted from the optimal state values.

Analyzing the Optimal Policy Resulting from RL

5) We are asked to plot the optimal policy of the agent navigating the 2-D state-space. My plot is presented below in **Fig. 4**

Fig. 4
Optimal Policy: Reward Function 1



Observations

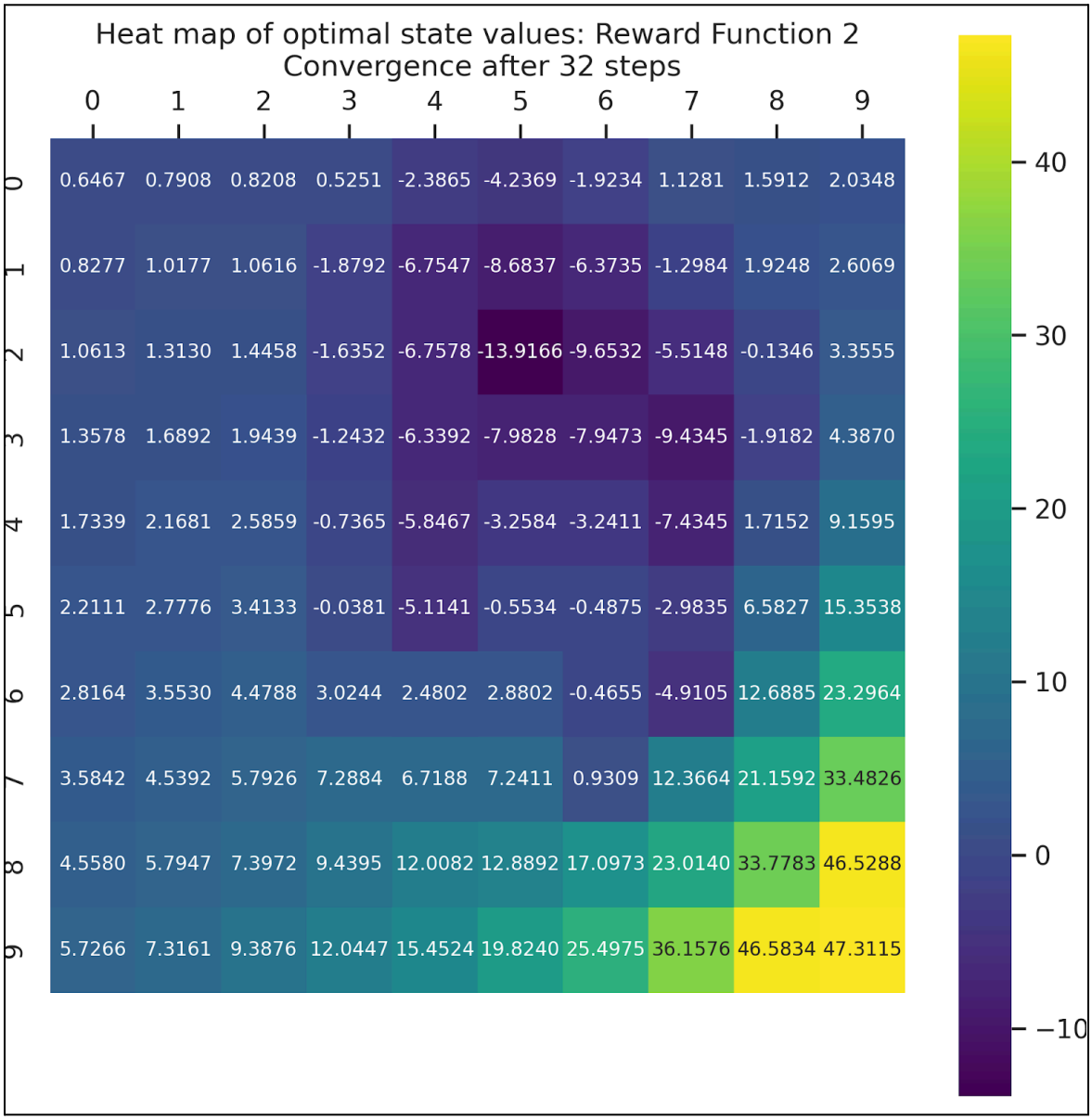
The optimal policy (presented above in Fig. 4) does match my intuition. Following the policy from any state eventually leads to state 99, which is the only state with a positive reward and, therefore, the goal. Even the policy from state 99, which has the agent staying in place, moves the agent towards the goal.Therefore, the agent can compute the optimal action to take at each state by observing the optimal values of its neighboring states - as we have done to derive the optimal policy presented above

Repeating the Process for Reward Function 2

6) We are asked to repeat the process we followed for question two but this time to use **Reward Function 2** instead of Reward Function 1. For reference, both reward functions are presented above in Fig. 1.

We are asked to plot the Optimal State Value matrix (after convergence). My plot is presented below in Fig. 5

Fig. 5
Optimal State Values: Reward Function 2
Convergence After 32 Steps



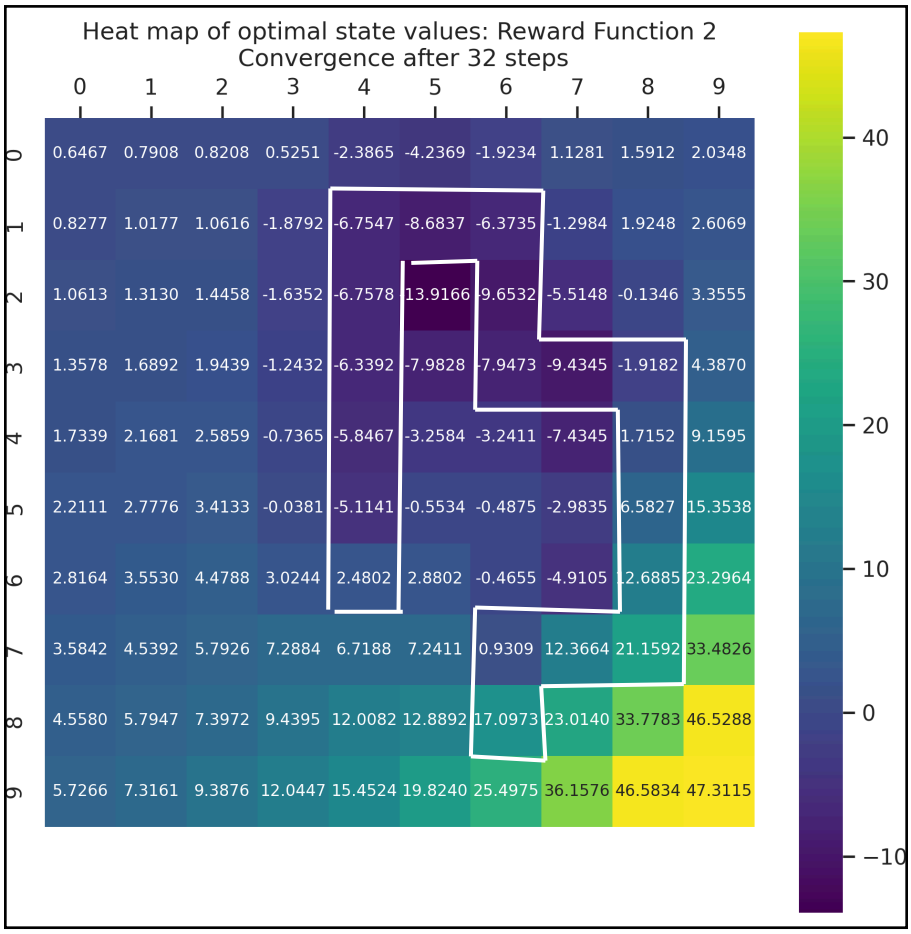
7) We are also asked to present the optimal state value matrix as a heat map which I have done above in **Fig. 5** We can make several observations from the **heat map** presented above **in Fig. 5.**

- The state with the brightest color - state 99 - is also the state with the highest reward (10). Furthermore, state 99 is the only state with a positive reward, which makes it a clear goal. In the heat map, we observe that the closer a state is to state 99, the brighter it is, representing a higher optimal value.

- While the state with the darkest color, state 25, is not one of the states with the lowest reward values (-100), it is surrounded by states with the lowest reward values, making state 25 a dead-end state. Intuitively, this state would be one of the least desirable states to wind up in.

- In Reward Function 2, the states with the lowest (and negative) reward values (-100) are connected and wrapped around 11 states in an enclosed space with one way in and out. In Fig. 6 below, I have drawn a white line around these negatively rewarded states. While most of these states have a negative value at step 32, those states closer to state 99 have a positive value, and the closest state (state 78) has the 10th highest score (21) despite a reward value of -100.

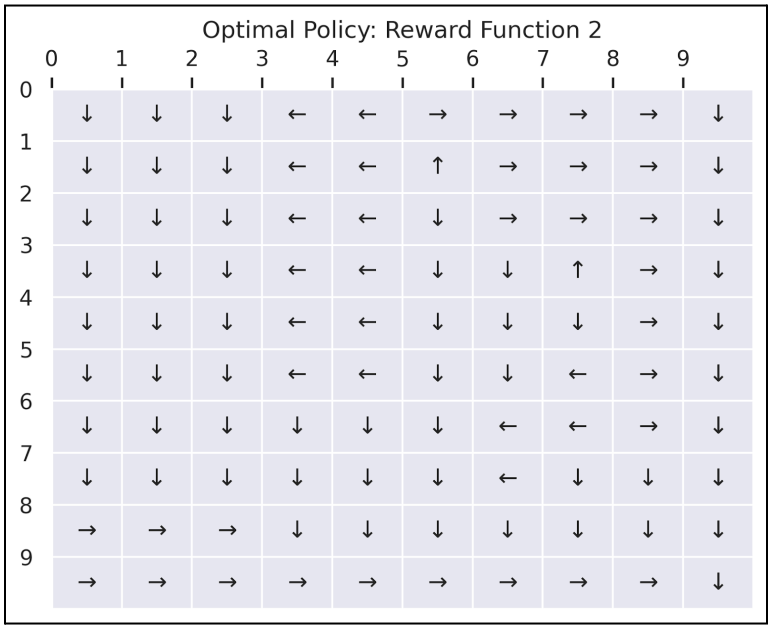
Fig. 6
Optimal State-Values after Convergence (step 32): RF2
With **Negatively Rewarded States outlined in White**



- Except for state 65, which sits at the entrance of the space enclosed by the negatively rewarded states, all other enclosed states have negative values at step 32, confirming that the enclosed space is undesirable.
- The above patterns suggest that the reward function could be extracted from the optimal state values.

8) We are asked to plot the optimal policy of the agent navigating the 2-D state-space with **Reward Function 2**. My plot is presented below in **Fig. 7**

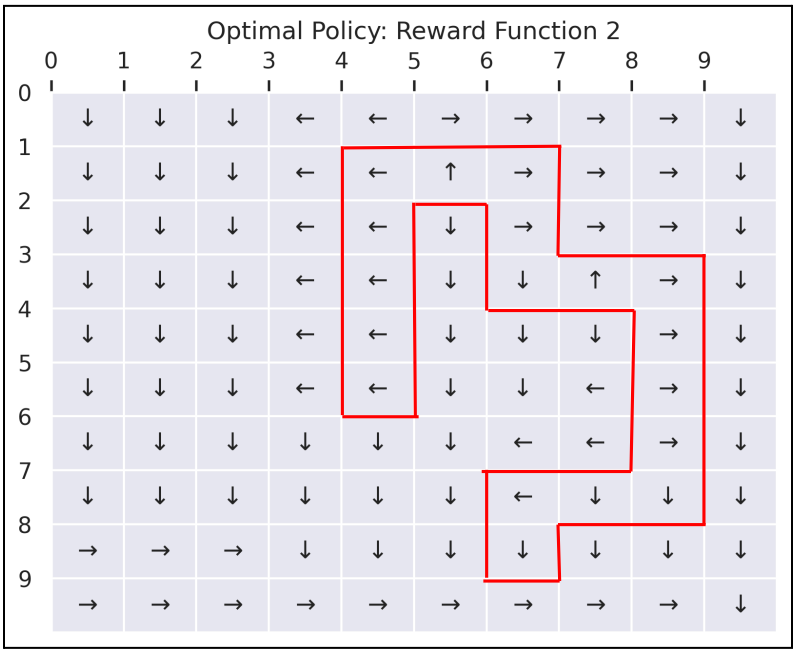
Fig. 7
Optimal Policy: Reward Function 2



Observations: The optimal policy (presented above in Fig. 7) matches my intuition in the following ways:

- The policy from any state eventually leads to state 99, which is the only state with a positive reward and, therefore, the goal. Even the policy from state 99, which has the agent staying in place, moves the agent towards the goal.
- Intuitively I would expect the agent to avoid wandering into the enclosed space created by the negatively rewarded states as this space represents a dead-end for the agent who is trying to avoid negative rewards. In Fig. 8 below, I have outlined the negatively rewarded states in red. With the exception of states 80-82, no states have a policy pointing the agent in the direction of the dead-end space

Fig. 8
Optimal Policy: **Reward Function 2**
With Negatively Rewarded States outlined in Red

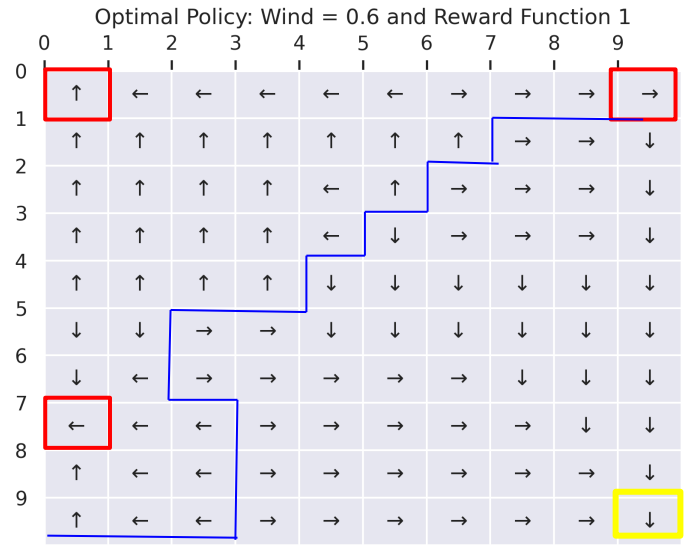


Examining the Effect of Modifying the Hyper-Parameter w

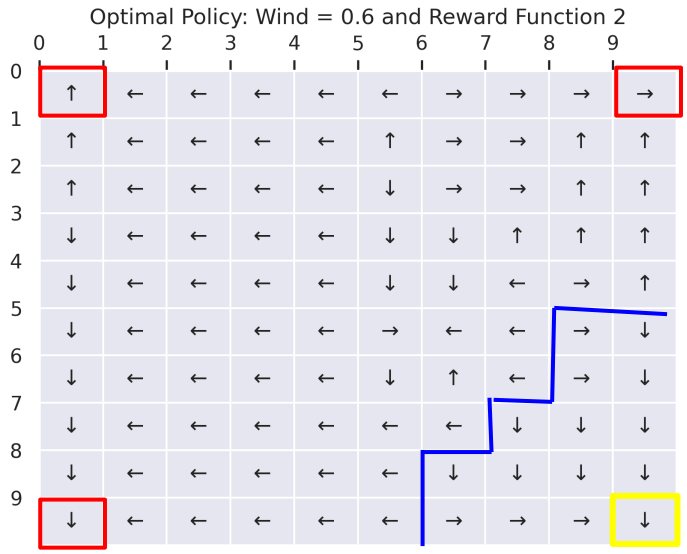
9) In this part, we change the hyper-parameter w (representing the wind) from 0.1 to 0.6 and then repeat the Reinforcement Learning process from above to generate the optimal policy using **Reward Function 1** and again using **Reward Function 2**. My results are presented below in Fig 7 along with my prior results with $w = 0.1$ for comparison purposes.

Fig 9
Effect of increasing hyper-parameter w from 0.1 to 0.6

Reward Function 1



Reward Function 2



Observations:

The optimal policies generated for both reward functions with the wind set to 0.6 are problematic in two ways. First of all, in each policy, three states (outlined in red) instruct the agent to leave the grid. Furthermore, the optimal policy paths from many states lead to these “leave-grid” states.

On each policy, I have overlaid a blue boundary dividing the states into two groups. The states to the right of the boundary are those from which the optimal policy leads the agent to the goal state (outlined in yellow.) The states to the left of the boundary are those from which the optimal policy leads the agent to one of the ‘leave-grid’ states (outlined in red). The optimal policy for Reward Function 1 leads the agent off-grid from roughly half of the states. The situation is worse for the optimal policy for Reward Function 2, which leads the agent off the grid for 85/100 states.

Based on the results presented above in Fig. 9, w=0.1 is the preferred setting, which I will use for the remainder of the project

Part 2: Inverse Reinforcement Learning (IRL)

Rewriting the LP formulation Equation using Block Matrices

10) As noted in the project documentation, the linear programming (LP) formulation of the inverse reinforcement learning (IRL) algorithm is given as follows:

LP Formulation
Statement Set 1

We seek to **maximize:** $R,\ t_i,\ u_i$ subject to the following:

Statement #	Equation	Conditions
1a	$\sum_{i=1}^{\lvert s\rvert} (t_i - \lambda u_i)$	
1b	$[(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1})^{-1} R] \geq t_i$	$\forall a \in A \backslash a_1,\ \forall i$ *For all a in A except a_1 and for all i*
1c	$(P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1} R] \succcurlyeq 0$	$\forall a \in A \backslash a_1$ *For all a in A except a_1*
1d	$-u \preccurlyeq R \preccurlyeq u$	
1e	$\lvert R_i\rvert \leq R_{max}$	$i = 1, 2, \ldots, \lvert s\rvert$ *For i in the set of integers from 1 to the number of states.*

Our task in Q10 is to figure out how to represent the equations above in a the form understood by the silver.lp function which we can then call to minimize $-(R, t_i, u_i)$ thus giving us the solution that maximizes R, t_i, u_i. Then we can extract the reward function R from this solution.

Therefore, we want to represent **Statement Set A** (above) as the equivalent **Statement Set B** (presented on the following page)

LP Formulation
Statement Set 2

We seek to **maximize:** x subject to the following:

Eq #	Equation	Conditions
2a	$c^T x$	
2b	$Dx \preccurlyeq b$	$\forall a \in A \backslash a_1$ *For all a in A except a_1*

Step 1: Based on the fact that **Statement Set 1** seeks to maximize R, t_i, u_i and **Statement Set 2** seeks to maximize x, we know that:

$$x = \begin{bmatrix} t \\ u \\ R \end{bmatrix}$$

Step 2: Statement 1a and Statement 2a are equivalent, so:

$$\sum_{i=1}^{|s|} (t_i - \lambda u_i) = c^T x$$

But, the left side of the equation in matrix form is just $t - \lambda u$, so we need to find a c such that

$$c^T \cdot \begin{bmatrix} t \\ u \\ R \end{bmatrix} = t - \lambda u$$

Therefore:

$$c = \begin{bmatrix} 1 \\ -\lambda \\ 0 \end{bmatrix}$$

Step 3: Next we need to express Statement 2b (which represents the conditions of our linear program) in terms of the conditions in Statement Set 1 - Statements 1b-1e.

Specifically, we need to express matrices D, x, and b in terms of Pa_1, P_a, t, u, R, and R_{max} so that statement 2b:

$$Dx \preccurlyeq b$$

represents Statements 2a-5a. We will handle statements 2a-5a one at a time and build our D and b matrices from top to bottom with the top lines corresponding to statement 2 and the bottom lines corresponding to statement 5a

Add statement 1b:

$$[(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1})^{-1}R] \geq t_i$$

Rewriting the condition in matrix form yields:

$$[(P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1}R] \geq t$$

The solver.lp function expects conditions in the form $statement \leq value$ and since t is part of x we need to rewrite the condition as:

$$t - (P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1}R] \preccurlyeq 0$$

Adding this condition to our D and b matrices gives us:

$$\begin{bmatrix} \mathbf{I}_{|s|\,x\,|s|} & \mathbf{0}_{|s|\,x\,|s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s|\,x\,|s|} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} t \\ u \\ R \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{|s|\,x\,|s|} \\ \vdots \end{bmatrix}$$

D x b

Since P_a represents the three non-optimal actions, we need to repeat the top rows three times - once for each non-optimal action. So our matrices become

$$\begin{bmatrix} \mathbf{I}_{|s|\,x\,|s|} & \mathbf{0}_{|s|\,x\,|s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s|\,x\,|s|} \\ \mathbf{I}_{|s|\,x\,|s|} & \mathbf{0}_{|s|\,x\,|s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s|\,x\,|s|} \\ \mathbf{I}_{|s|\,x\,|s|} & \mathbf{0}_{|s|\,x\,|s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s|\,x\,|s|} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} t \\ u \\ R \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{|s|\,x\,|s|} \\ \mathbf{0}_{|s|\,x\,|s|} \\ \mathbf{0}_{|s|\,x\,|s|} \\ \vdots \end{bmatrix}$$

D x b

Add statement 1c:

$$[(P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1}R] \geq 0$$

Rewriting the inequality in $statement \preccurlyeq value$ form gives us

$$-(P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1}R] \preccurlyeq 0$$

Since P_a appears in the condition, we need to add three rows (one for each non-optimal action to our matrices yielding:

$$\underbrace{\begin{bmatrix} I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ 0_{|s| \times |s} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ \vdots & \vdots & \vdots \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} t \\ u \\ R \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 0_{|s| \times |s|} \\ 0_{|s| \times |s|} \\ 0_{|s| \times |s|} \\ 0_{|s| \times |s} \\ 0_{|s| \times |s} \\ 0_{|s| \times |s} \\ \vdots \end{bmatrix}}_{b}$$

Add statement 1d:

$$-u \preccurlyeq R \preccurlyeq u$$

Rewriting the condition as two inequalities gives us:

$$-u \preccurlyeq R \text{ and } R \preccurlyeq u$$

Since R and u are both part of x we want them on the left side of the inequality, so we need to rewrite our inequalities as:

$$-u - R \preccurlyeq 0 \text{ and } -u + R \preccurlyeq 0$$

Adding these two conditions to our matrices yields:

$$\underbrace{\begin{bmatrix} I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ 0_{|s| \times |s} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}{}_{|s| \times |s|} \\ 0_{|s| \times |s|} & -I_{|s| \times |s|} & -I_{|s| \times |s} \\ 0_{|s| \times |s|} & -I_{|s| \times |s|} & I_{|s| \times |s} \\ \vdots & \vdots & \vdots \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} t \\ u \\ R \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 0_{|s| \times |s|} \\ 0_{|s| \times |s|} \\ 0_{|s| \times |s|} \\ 0_{|s| \times |s} \\ 0_{|s| \times |s} \\ 0_{|s| \times |s} \\ 0_{|s| \times |s} \\ 0_{|s| \times |s} \\ \vdots \end{bmatrix}}_{b}$$

Add statement 1e:

$$|R_i| \leq R_{max}$$

Removing the absolute value sign from the statement yields the following two inequalities:

$$-R \preccurlyeq R_{max} \quad \text{and} \quad R \preccurlyeq R_{max}$$

Adding these two conditions to our matrices yields our final representation:

$$\begin{bmatrix} I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}_{|s| \times |s|} \\ I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}_{|s| \times |s|} \\ I_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}_{|s| \times |s|} \\ 0_{|s| \times |s|} & -I_{|s| \times |s|} & -I_{|s| \times |s|} \\ 0_{|s| \times |s|} & -I_{|s| \times |s|} & I_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & -I_{|s| \times |s|} \\ 0_{|s| \times |s|} & 0_{|s| \times |s|} & I_{|s| \times |s|} \end{bmatrix} \begin{bmatrix} t \\ u \\ R \end{bmatrix} = \begin{bmatrix} 0_{|s| \times |1|} \\ 0_{|s| \times |1|} \\ 0_{|s| \times |1|} \\ 0_{|s| \times |1|} \\ 0_{|s| \times |1|} \\ 0_{|s| \times |1|} \\ 0_{|s| \times |1|} \\ 0_{|s| \times |1|} \\ R_{max}{}^{|s| \times |1|} \\ R_{max}{}^{|s| \times |1|} \end{bmatrix}$$

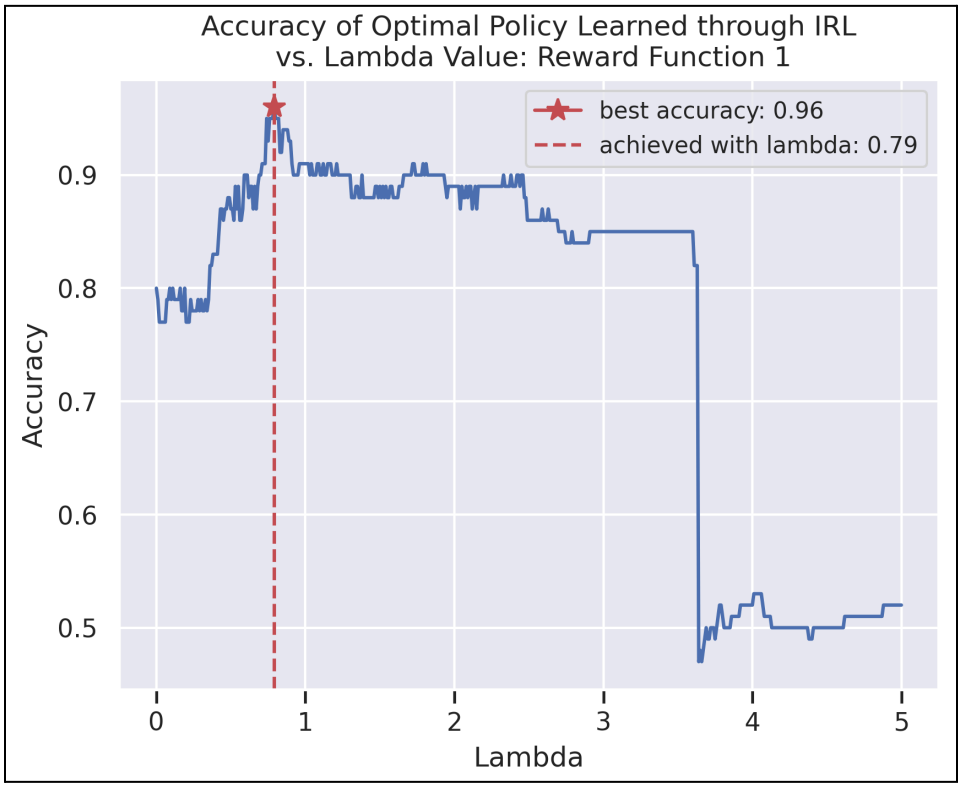
D — $10|s| \times 3|s|$ $\quad$ x — $3|s| \times 1$ $\quad$ b — $10|s| \times 1$

Performing IRL with Reward Function 1

11) In this part we implement an inverse reinforcement learning algorithm whereby we seek to extract (or learn) an expert's reward function from their optimal policy. Then we use this extracted (learned) reward function to learn the optimal value-states and policy through the reinforcement learning (RL) algorithm implemented in part 1.

Our task in question 11 is to implement the IRL algorithm while also tuning the λ hyper-parameter. We swept λ from 0 to 5 over 500 evenly spaced values, executed the IRL algorithm and then compared the accuracy of the resulting agent's optimal policy by comparing it to the expert's optimal policy that we used to extract the reward function. We define accuracy as the number of matching state actions divided by the total number of state actions. Finally, we plot optimal policy accuracy vs. lambda-value. My results are presented below in **Fig. 8**

Fig. 10
Accuracy of IRL Agent Optimal Policy vs. Lambda Value
Reward Function 1

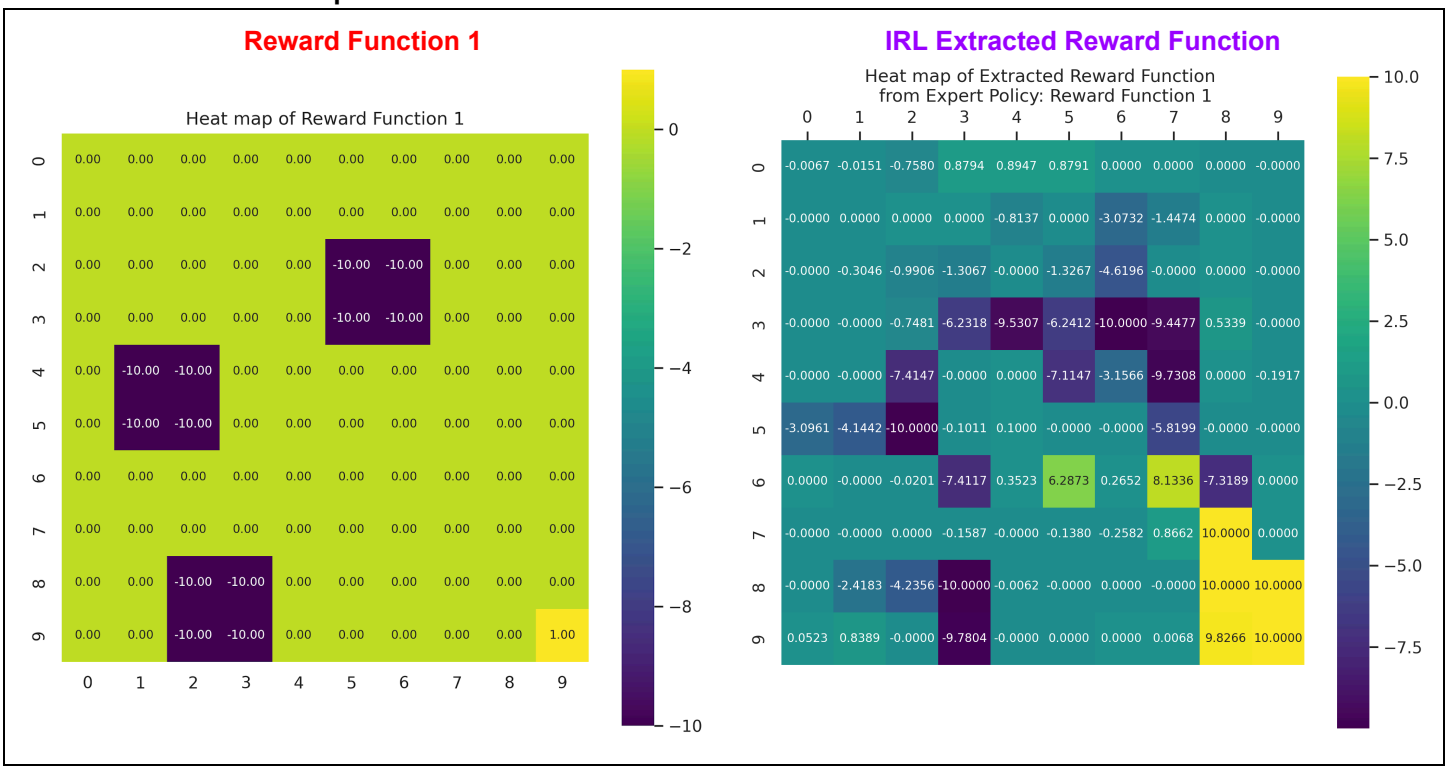


12) The goal of question 11 is to tune the λ hyper-parameter to the IRL algorithm. Using the results presented above in Fig. 8, we identified the optimal λ value for IRL using Reward Function 1 (or $\lambda_{max}^{(1)}$) as follows:

$$\lambda_{max}^{(1)} = 0.79$$

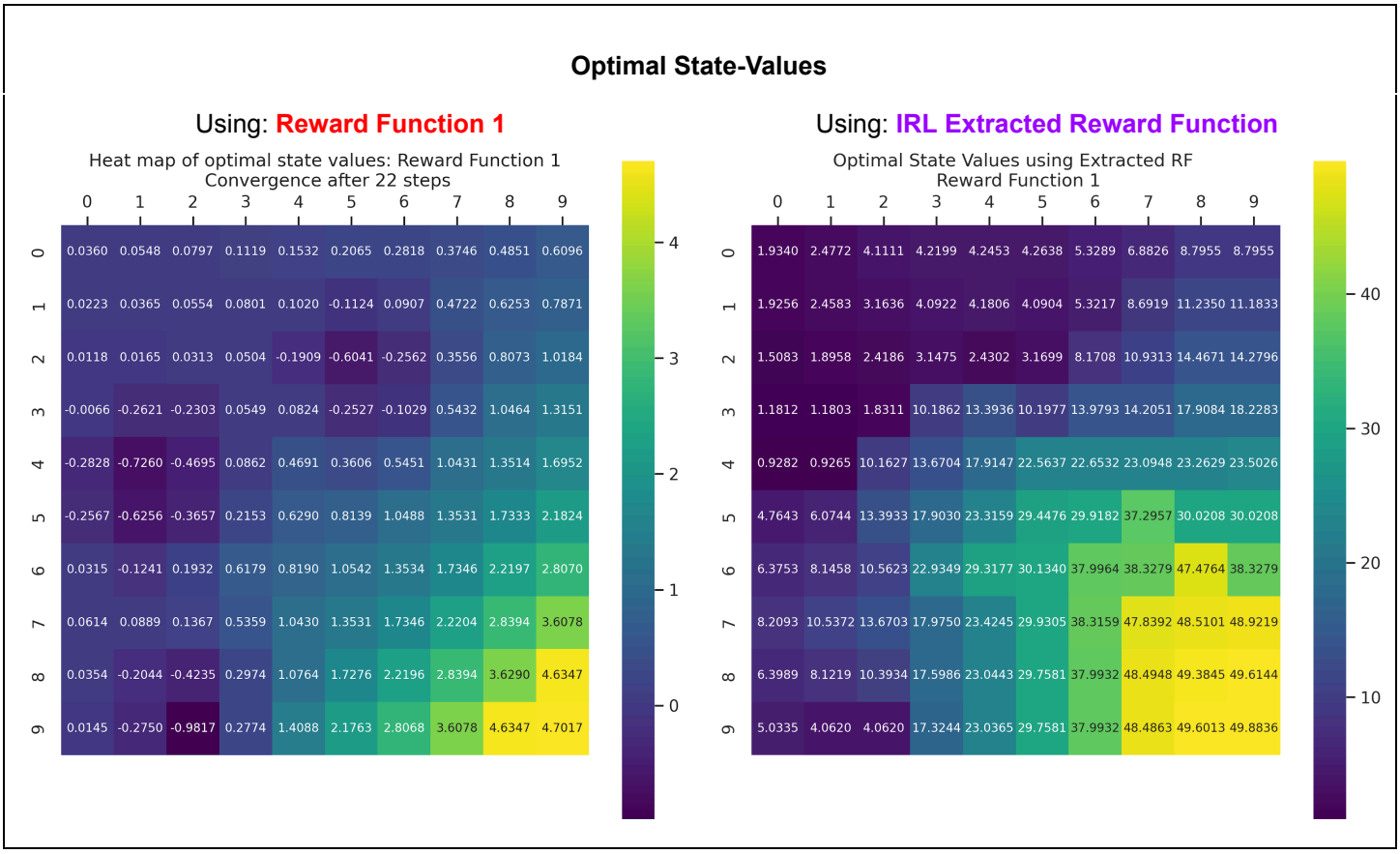
13) Using the optimum lambda value for Reward Function 1 discovered above $\lambda_{max}^{(1)} = 0.80$, we can now configure our IRL algorithm with this lambda value and rerun it to extract the optimal expert's reward function. For visualization and comparison purposes we are asked to plot the heatmap of this extracted reward function along the heat map of Reward Function 1. These two plots are presented below in **Fig. 9**

Fig. 11
Comparison of Reward Function 1 with IRL Extracted Reward Function



14) For this question, we use the IRL extracted reward function (displayed above in Fig. 9) to train the agent and obtain the optimal state-values. For visualization and comparison purposes we are asked to plot a heatmap of the IRL trained optimal state-values along with the heat map of the Expert Optimal State-Values (obtained in Q5). These two plots are presented below in **Fig. 10**

Fig. 12
Comparison of optimal state-values obtained with:
Reward Function 1 vs IRL Extracted Reward Function

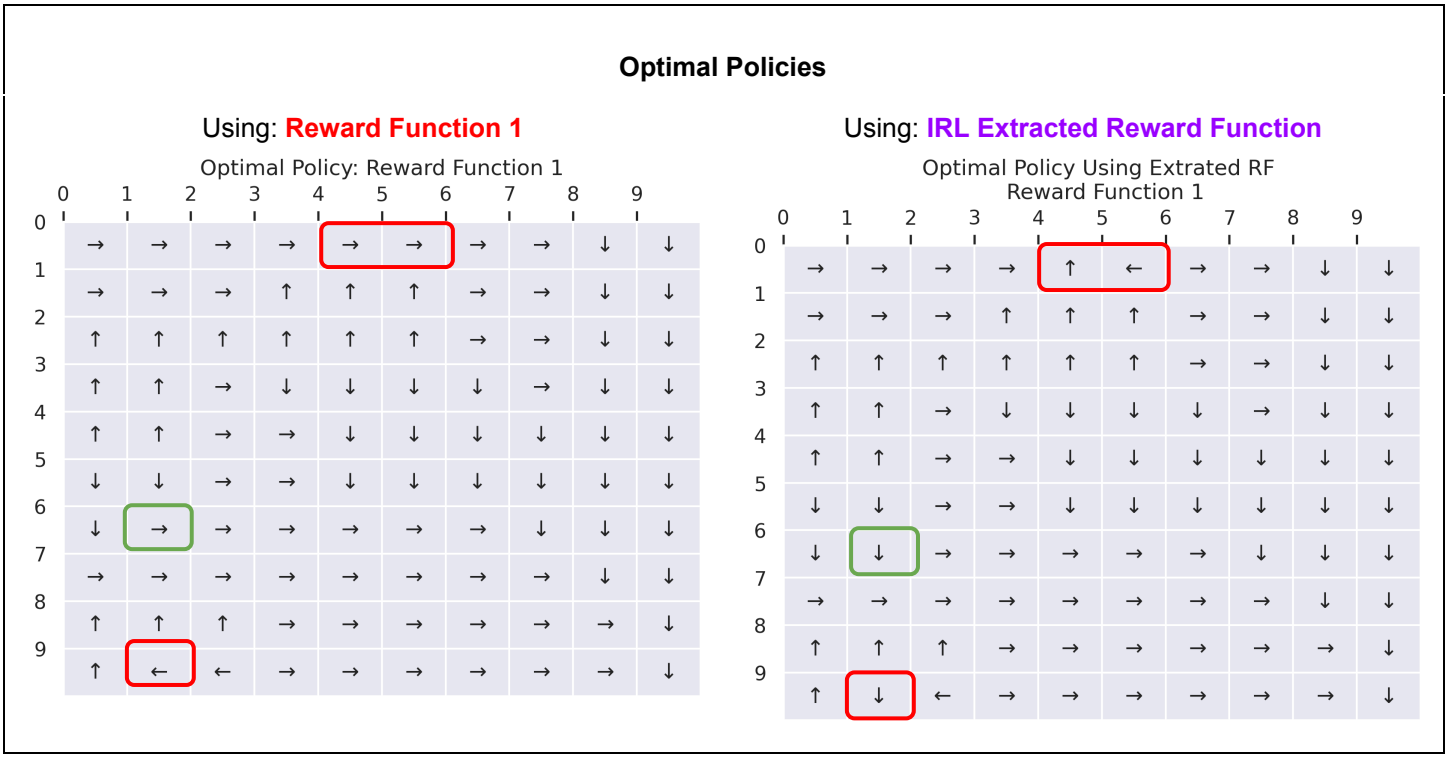


15) In this question we are asked to compare the two plots above in **Fig. 12**. Here are my observations:

- **Reward Function 1** produces optimal state values that include negative values, while the **IRL Extracted Reward** only produces positive state values. That said, both reward functions produce optimal state values within a similar range, with the highest value being ~ 4.7 for **Reward Function 1** and ~5.0 for the **IRL Extracted Reward Function** and the lowest value being ~ -1 for Reward Function 1 and ~ 0 for the IRL Extracted Reward Function.
- In both heat maps, the brightest color and highest value belong to state 99 (the goal state). The closer a state is to the goal state, the brighter it is, with brighter colors representing higher optimal state values. This creates a similar pattern in both heat maps, whereby states become brighter as they move toward the bottom right corner.
- In both heat maps, there is a pattern radiating from state 99 toward state 0, whereby the states get darker as you move from state 99 towards state 0, but the shape differs. In the **Reward Function 1** generated heat map, the shape is seen as diagonal lines of similar colors, while in the **IRL Extracted Reward Function** generated heat map, the shape is square-like. This difference can be attributed to the difference in the reward functions. In **Reward Function 1**, only state 99 has a positive reward value, while in the extracted reward function, there are five states clumped together in the bottom right corner that all share relatively the same highest reward value. These five states form a shape similar to the one we observe in the optimal state values heat map for the **IRL Extracted Reward Function**.

16) In this question we are asked to use the IRL extracted Reward Function to compute the optimal policy for Reward Function 1 and then plot it. My plot is presented below in **Fig. 13.** For comparison purposes, I have also included the Optimal Policy generated using Reward Function 1

Fig. 13
Comparison of optimal policies obtained with:
Reward Function 1 vs IRL Extracted Reward Function

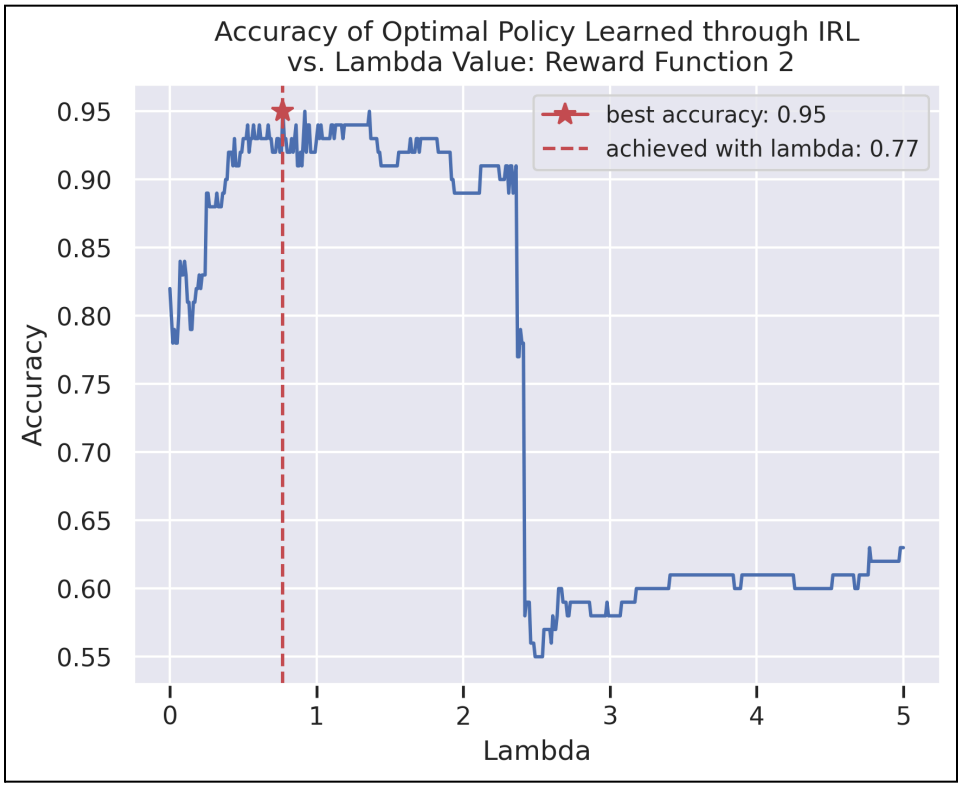


17) The two optimal policies shown above in **Fig. 13** are almost identical. They only differ in 4 states, which I have outlined. The state outlined in green represents a difference that doesn't affect the agent's ability to navigate to the goal state effectively. On the other hand, the differences in the states outlined in red affect the agent's ability to effectively navigate to the goal in the policy produced from the Extracted Reward Function. States 40 and 19 instruct the agent to move off the grid, and state 50 instructs the agent to move to state 40.

Performing IRL with Reward Function 2

18) In this part we repeat the process for question 11, but we use Reward Function 2 instead of Reward Function 1. As in question 11, our tsk in question 18 is to implement the IRL algorithm while also tuning the λ hyper-parameter. We swept λ from 0 to 5 over 500 evenly spaced values, executed the IRL algorithm and then compared the accuracy of the resulting agent's optimal policy by comparing it to the expert's optimal policy that we used to extract the reward function. We define accuracy as the number of matching state actions divided by the total number of state actions. Finally, we plot optimal policy accuracy vs. lambda-value. My results are presented below in **Fig. 11**

Fig. 14
Accuracy of IRL Agent Optimal Policy vs. Lambda Value
Reward Function 2

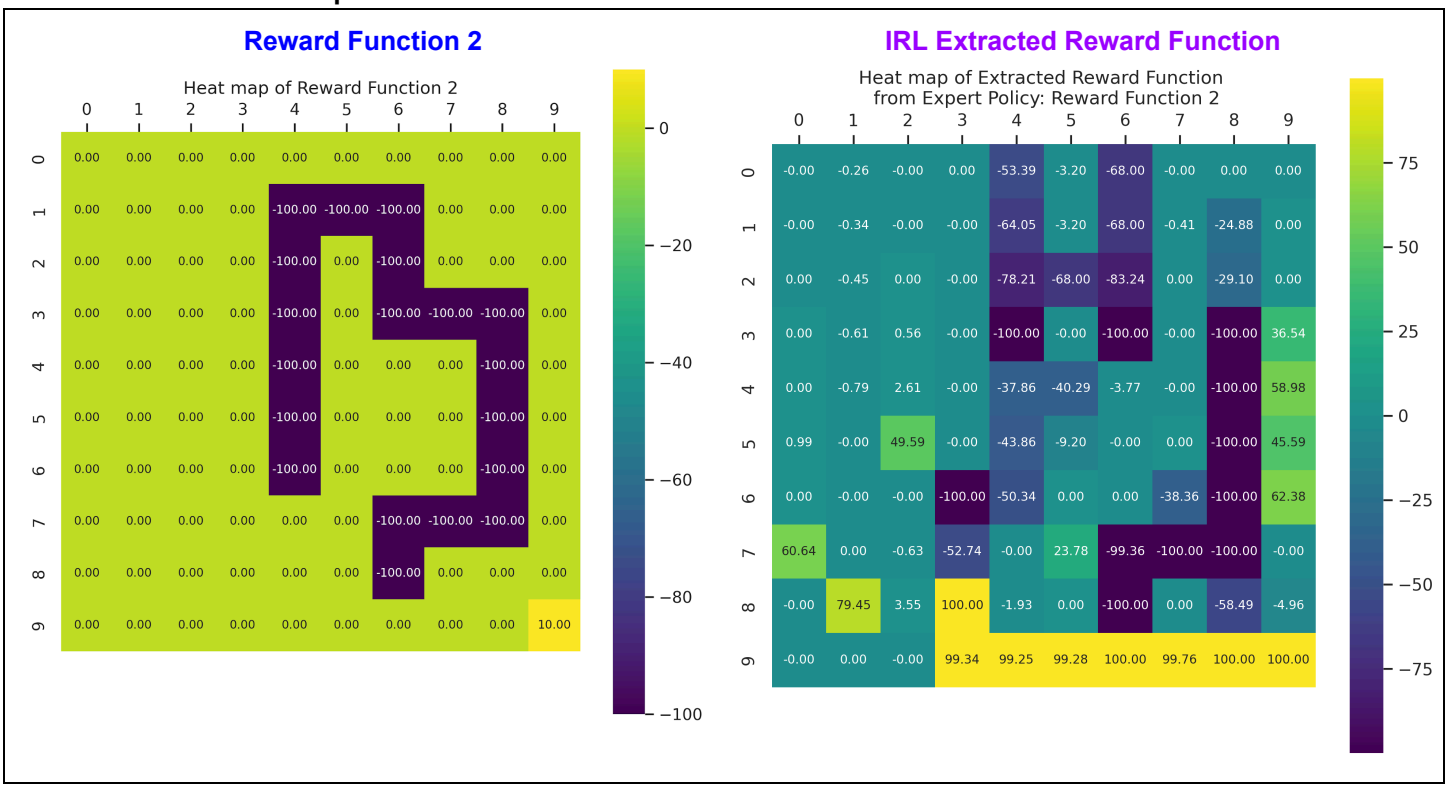


19) The goal of question 18 is to tune the λ hyper-parameter to the IRL algorithm. Using the results presented above in Fig. 14, we identified the optimal λ value for IRL using Reward Function 1 (or $\lambda_{max}^{(2)}$) as follows:

$$\lambda_{max}^{(2)} = 0.77$$

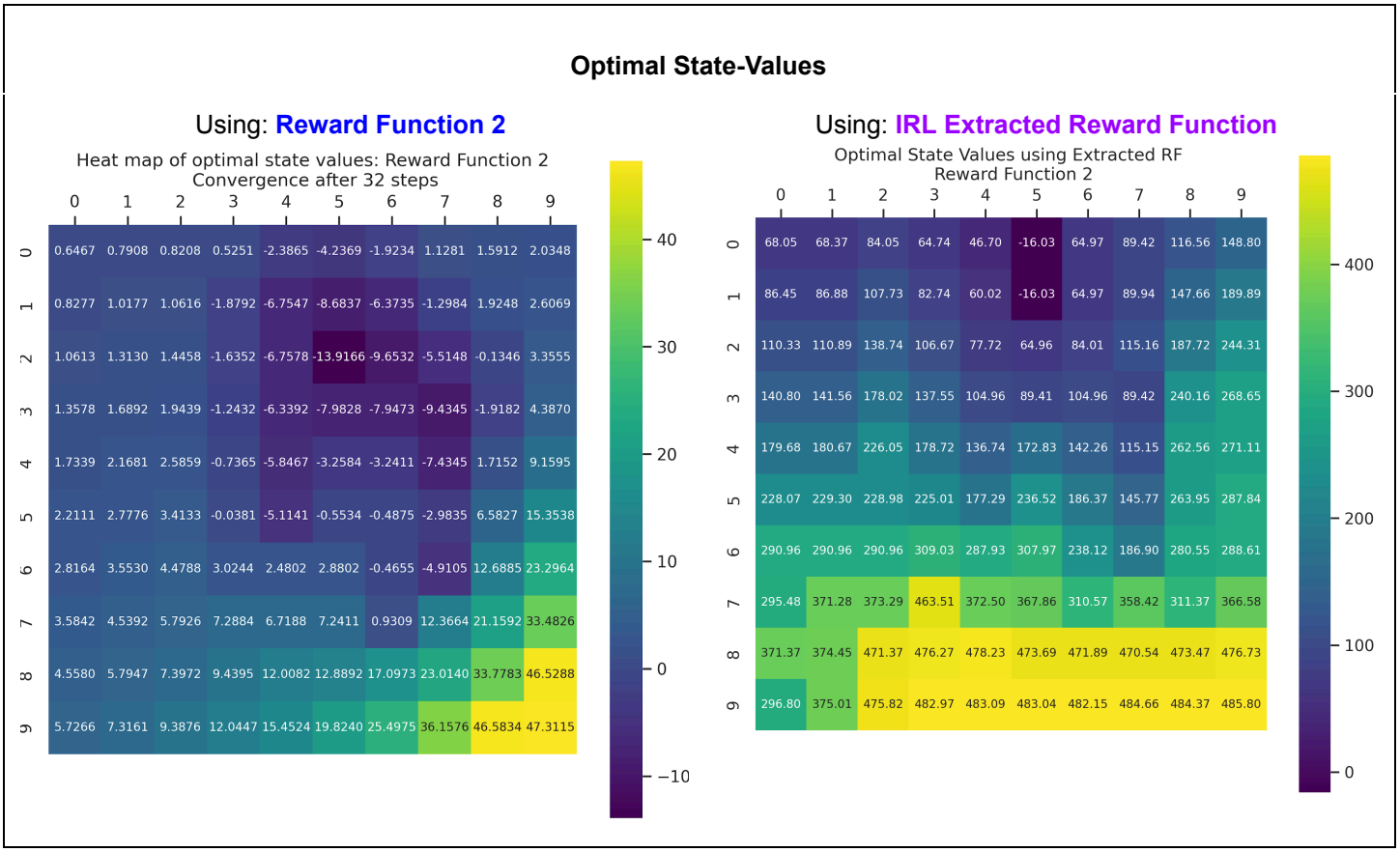
20) Using the optimum lambda value for Reward Function 2 discovered above $\lambda_{max}^{(2)} = 1.36$, we can now configure our IRL algorithm with this lambda value and rerun it to extract the optimal expert's reward function. For visualization and comparison purposes we are asked to plot the heatmap of this extracted reward function along the heat map of **Reward Function 2**. These two plots are presented below in **Fig. 12**

Fig. 15
Comparison of Reward Function 2 with IRL Extracted Reward Function



21) For this question, we use the IRL extracted reward function (displayed above in **Fig. 15**) to train the agent and obtain the optimal state-values. For visualization and comparison purposes we are asked to plot a heatmap of the IRL trained optimal state-values along with the heat map of the Expert Optimal State-Values (obtained in Q7). These two plots are presented below in **Fig. 136**

Fig. 16
Comparison of optimal state-values obtained with:
Reward Function 2 vs IRL Extracted Reward Function



22) In this question we are asked to compare the two plots above in **Fig. 16**

- Both reward functions produce optimal state values that include negative and positive values. However, the reward functions produce optimal state values that differ in their range by an order of magnitude. The highest value for **Reward Function 2** is ~ 47, while the highest for the **IRL Extracted Reward Function** is ~490. The lowest value for **Reward Function 2** is ~ -14, while the lowest for the **IRL Extracted Reward Function** is ~ -16.

- The brightest color and highest value in both heat maps belong to state 99 (the goal state). The closer a state is to the goal state, the brighter it is, with brighter colors representing higher optimal state values. This creates a similar pattern in both heat maps, whereby states become more colorful as they move toward the bottom right corner.

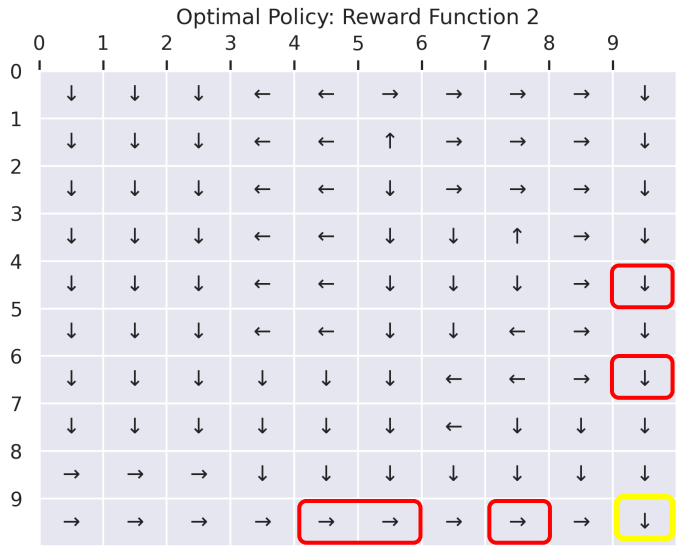
- In both heat maps, a pattern can be seen radiating from state 99, and the states get darker as you move away from state 99. However, in the **Reward Function 2** generated heat map, the radiation pattern is generally diagonal from the bottom right to the top left. Moving from state 99 diagonally towards the top left, the state tends to get darker. In the **IRL Extracted Reward Function**, the radiation pattern is generally from the bottom of the grid to the top.

23)

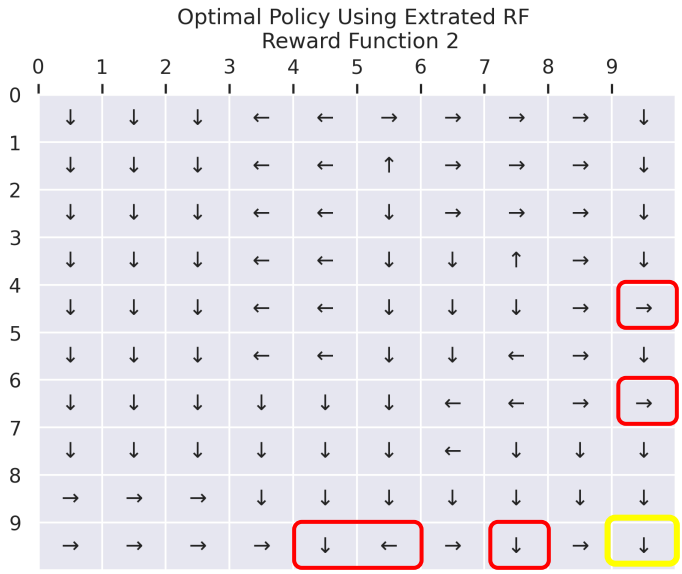
Fig. 17
Comparison of optimal policies obtained with:
Reward Function 2 vs IRL Extracted Reward Function

Optimal Policies

Using: **Reward Function 2**



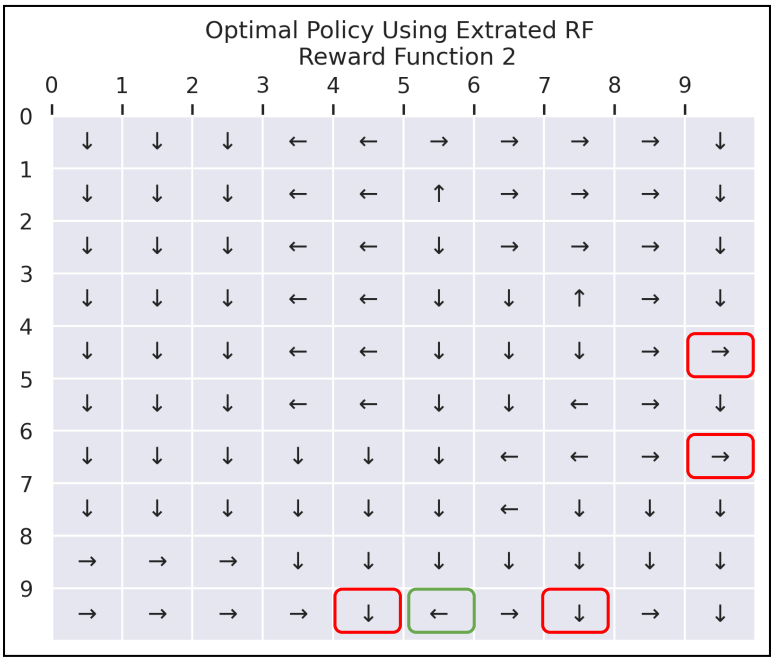
Using: **IRL Extracted Reward Function**



24) The two optimal policies shown above in **Fig. 17** are almost identical. They only differ in 5 states, which I have outlined in red. Unfortunately, all five of the differences affect the agent's ability to effectively navigate to the goal in the policy produced from the Extracted Reward Function. States 40, 70, 49, and 69 instruct the agent to move off the grid, and state 50 instructs the agent to move to state 40.

25) **Fig 18** below highlights **two discrepancies** in the Extracted Reward Function when starting with Reward Function 2.

Fig. 18
Discrepancies in IRL Extracted Reward Function
for Reward Function 2

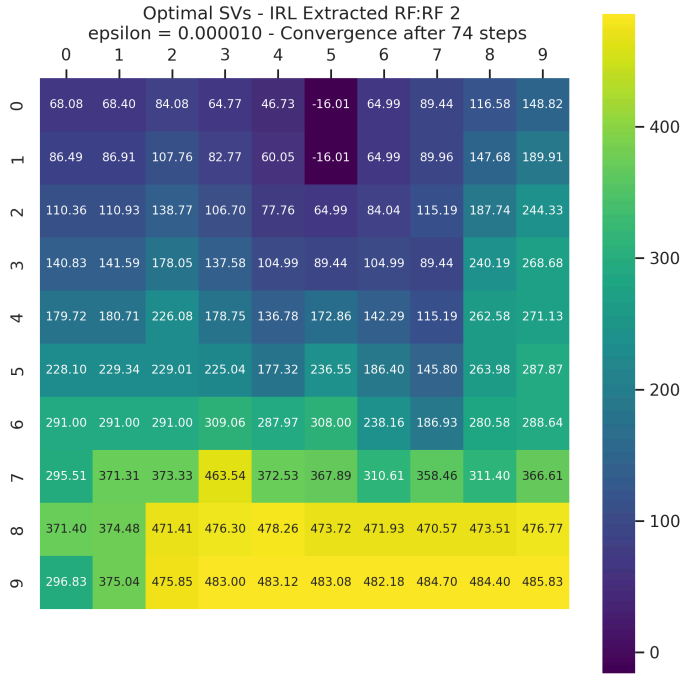


The first discrepancy involves the states outlined in red. The policy in these states instructs the agent to leave the grid. The reason for the discrepancy is likely the threshold (or epsilon value) in the value iteration algorithm which was set to 0.01. If we observe the optimal values at these states (shown above in **Fig. 16**), we notice that they are very close to the values at their neighboring states. To fix the discrepancy I adjusted the epsilon value from 0.01 to 1e-5 in the value iteration algorithm and re-ran the algorithm to generate the optimal state values and from there the optimal policy. My results with the new epsilon are displayed on the next page in **Fig. 19. As seen in Fig. 19, adjusting the epsilon value in the value iteration algorithm did fix the discrepancy**

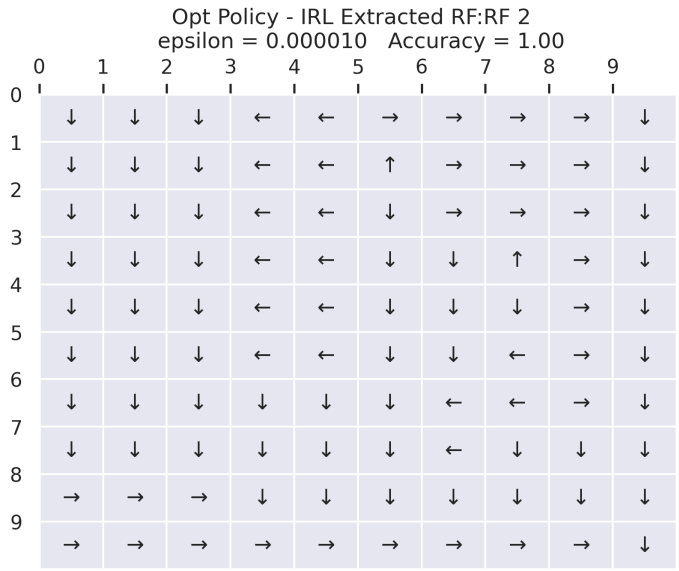
The second discrepancy involves the state outlined in green. The policy at this state is problematic because it directs the agent to a state which then instructs the agent to leave the grid. However, if the first discrepancy is fixed, this discrepancy is no longer problematic. **As seen in Fig. 19 on the next page, adjusting the epsilon value in the value iteration algorithm also fixed the second discrepancy**

Fig. 19
Optimal State Values and Policy with Extracted RF
Using Adjusted Epsilon Value = 1e-5

Optimal State Values



Optimal Policy



Accuracy vs. Lambda