# Project 3 - Reinforcement learning and Inverse Reinforcement learning

**Aryaman Gokarn**
**UID:506303588**

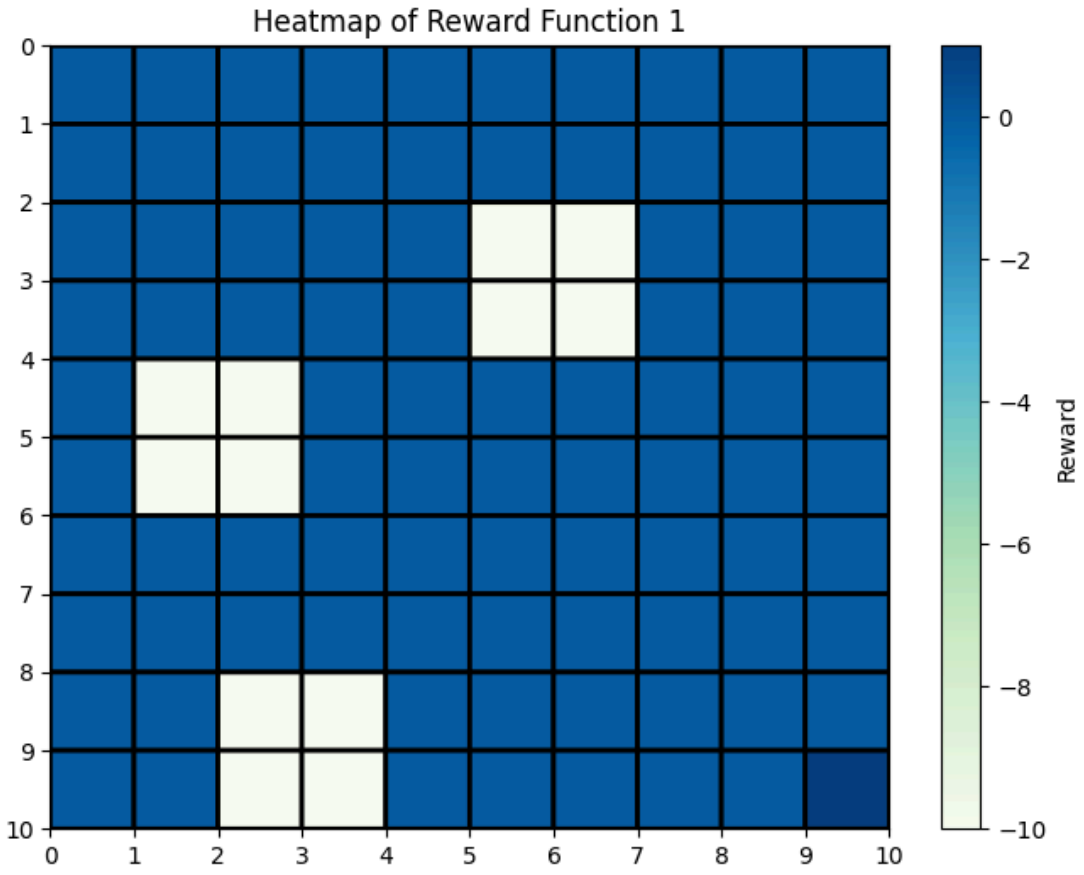
**Mugdha Bhagwat**
**UID: 606297799**

**Tania Rajabally**
**UID: 806153219**

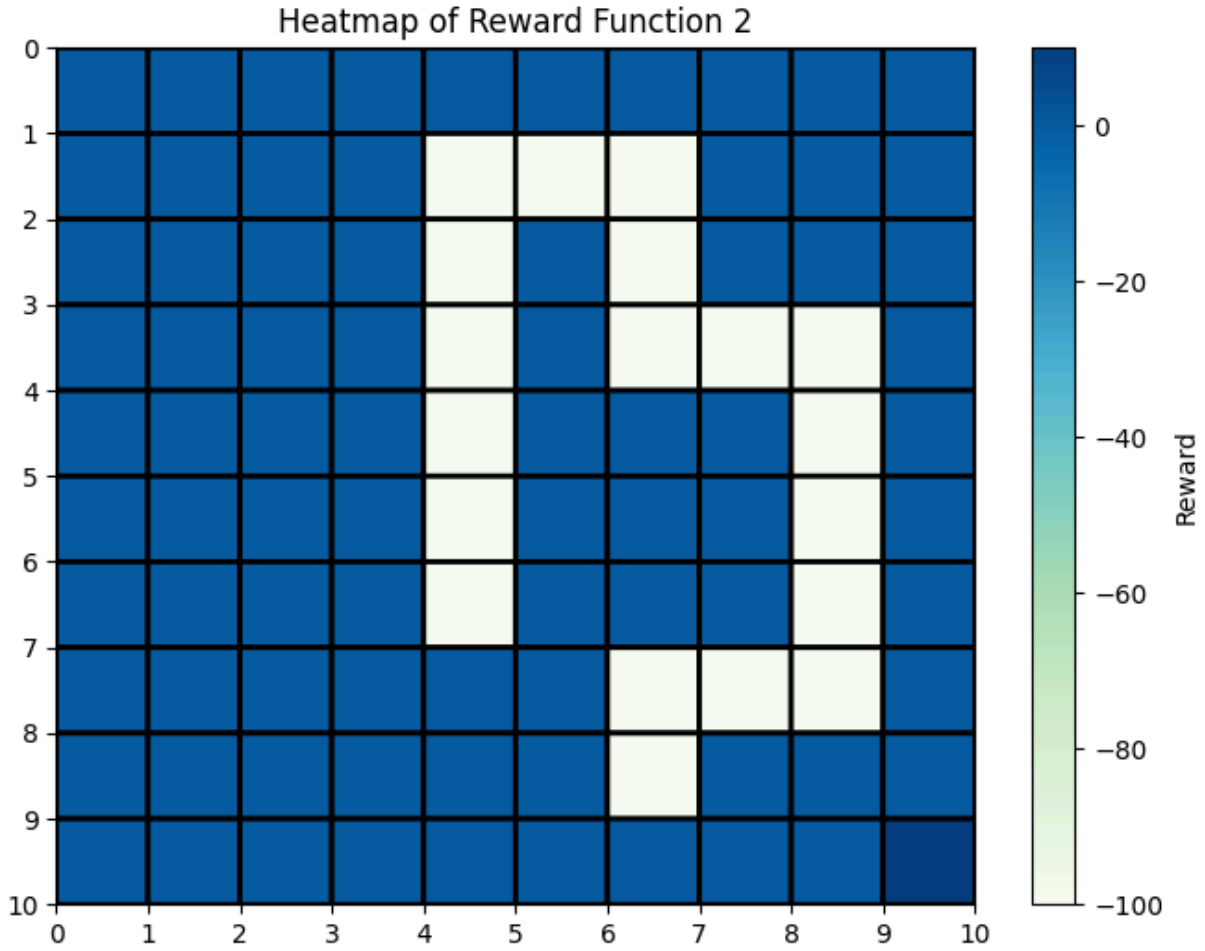
**QUESTION 1:**

**For visualization purpose, generate heat maps of Reward function 1 and Reward function 2. For the heat maps, make sure you display the coloring scale. You will have 2 plots for this question**

HeatMap for Reward Function 1:



HeatMap for Reward Function 2:




A heatmap is a graphical representation of data where the individual values contained in a matrix are represented as colors. We can refer to the scale on the side to see the corresponding reward values.

**QUESTION 2:**

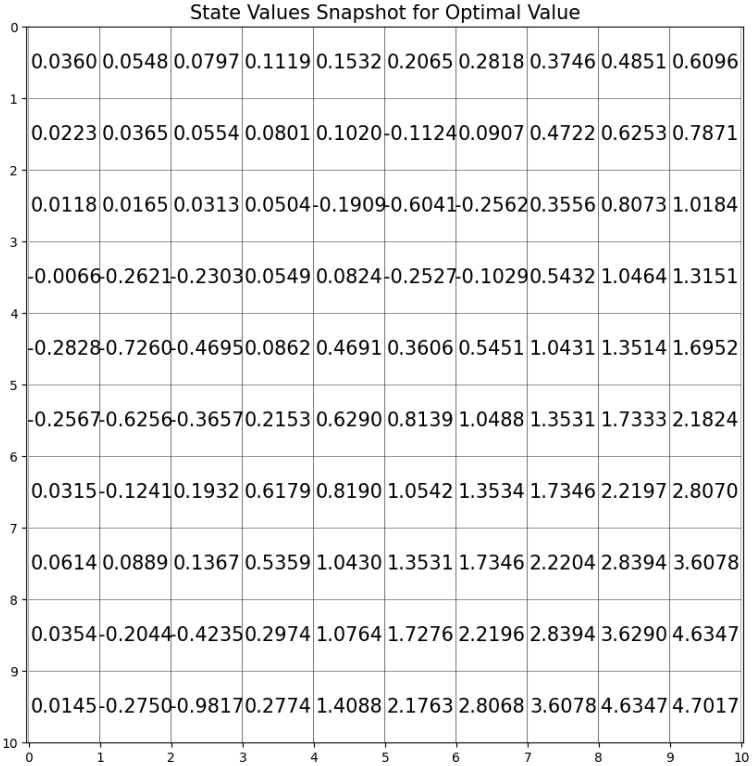
**Create the environment of the agent using the information provided in section 2. To be specific, create the MDP by setting up the state-space, action set, transition probabilities, discount factor, and reward function. For creating the environment, use the following set of parameters:**

- **Number of states = 100 (state space is a 10 by 10 square grid as displayed in figure 1)**
- **Number of actions = 4 (set of possible actions is displayed in figure 2)**
- **w = 0.1**
- **Discount factor = 0.8**
- **Reward function 1**

**After you have created the environment, then write an optimal state-value function that takes as input the environment of the agent and outputs the optimal value of each state in the grid. For the optimal state-value function, you have to implement the Initialization (lines 2-4) and Estimation (lines 5-13) steps of the Value Iteration algorithm. For the estimation step, use ε = 0.01. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal value of that state. In this part of question, you should have 1 plot.**

**Let's assume that your value iteration algorithm converges in N steps. Plot snapshots of state values in 5 different steps linearly distributed from 1 to N. Report N and your step numbers. What observations do you have from the plots?**

*Using the value iteration algorithm to get the optimal value of the state:*




From the above plot, we can see that the values are close to 0 i.e they are low in the initial states and then they start increasing as we reach the states with the maximum reward value.

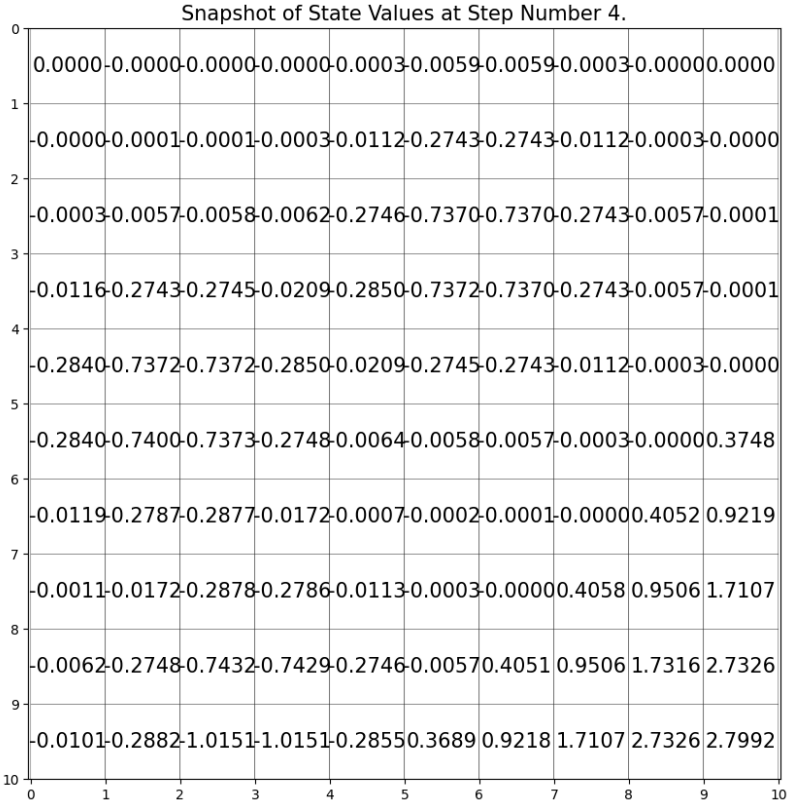
*Snapshots of state values in 5 different steps linearly distributed:*

Number of steps for Value Iteration Algorithm to converge: 22

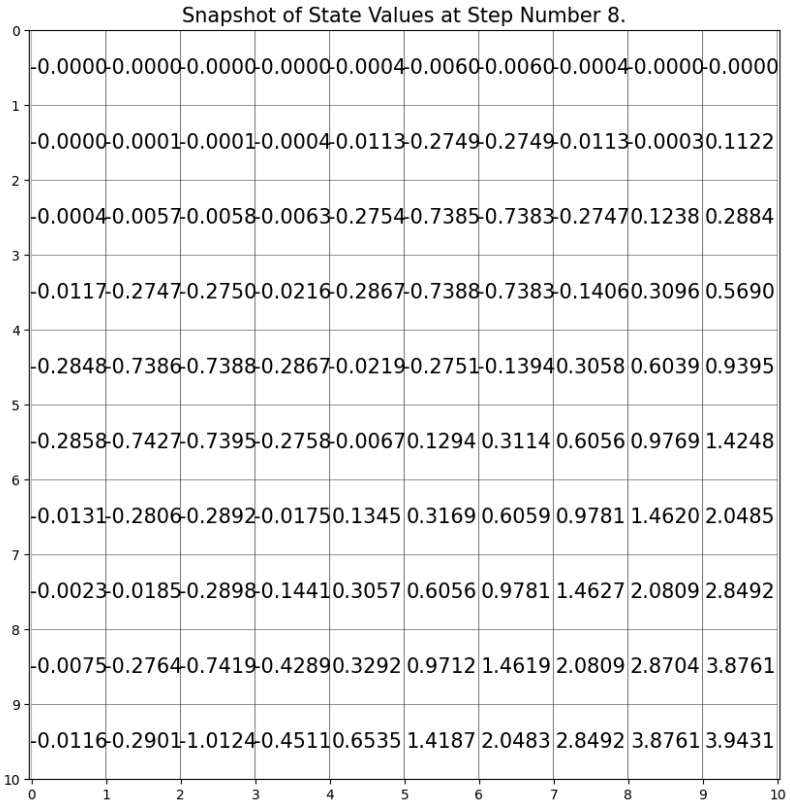
Number of intermediate steps captured: 5

The step numbers captured are: [4, 8, 12, 16, 20]

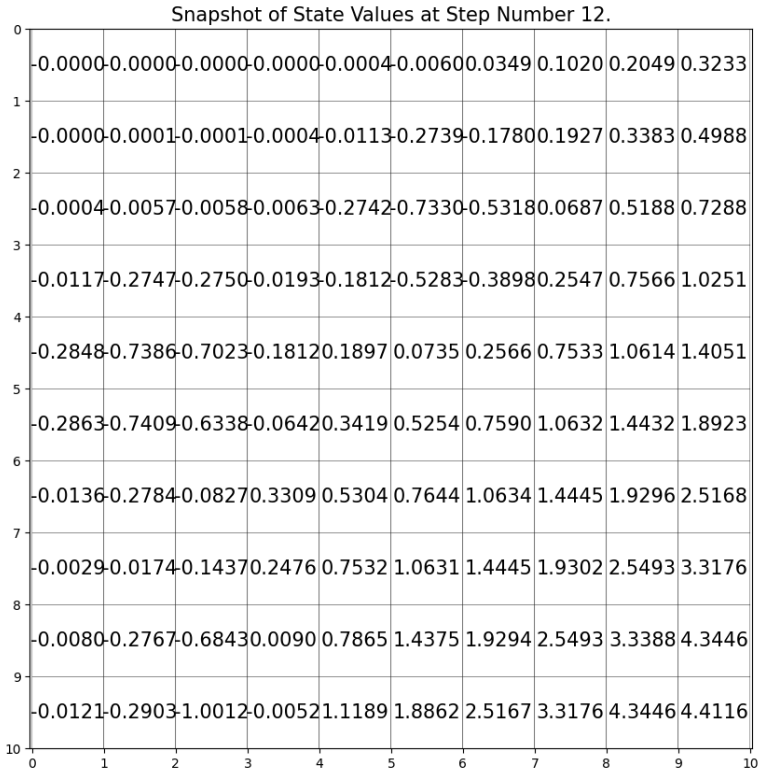
Step Number:  4
Value of Delta:  0.4844856000000002



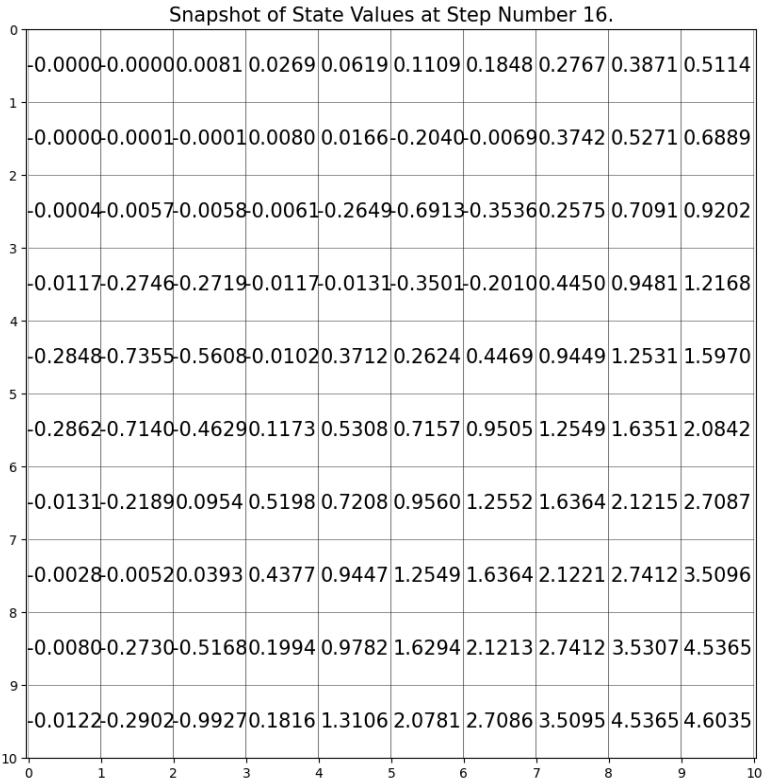

Step Number:  8
Value of Delta:  0.19838867471340826



Step Number:  12
Value of Delta:  0.08125961731799336




Step Number:  16
Value of Delta:  0.033283935823336996

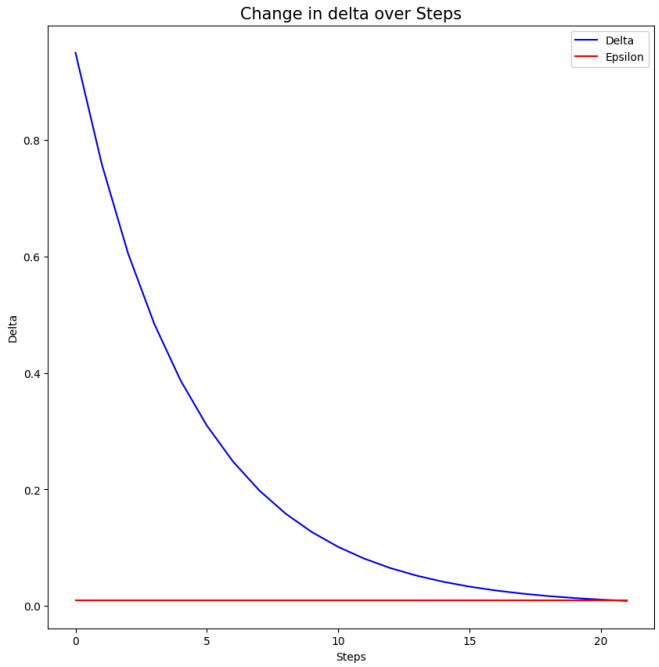


Step Number:  20
Value of Delta:  0.013633100077943716

Snapshot of State Values at Step Number 20.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0201 | 0.0370 | 0.0610 | 0.0926 | 0.1337 | 0.1869 | 0.2622 | 0.3550 | 0.4655 | 0.5899 |
| 1 | 0.0087 | 0.0205 | 0.0375 | 0.0614 | 0.0827 | -0.1319 | 0.0711 | 0.4525 | 0.6057 | 0.7675 |
| 2 | 0.0022 | 0.0031 | 0.0151 | 0.0323 | -0.2097 | -0.6235 | -0.2758 | 0.3360 | 0.7877 | 0.9988 |
| 3 | -0.0117 | -0.2718 | -0.2490 | 0.0354 | 0.0628 | -0.2723 | -0.1225 | 0.5236 | 1.0267 | 1.2955 |
| 4 | -0.2847 | -0.7320 | -0.4886 | 0.0666 | 0.4495 | 0.3409 | 0.5255 | 1.0235 | 1.3317 | 1.6756 |
| 5 | -0.2739 | -0.6445 | -0.3853 | 0.1956 | 0.6094 | 0.7943 | 1.0291 | 1.3335 | 1.7137 | 2.1628 |
| 6 | 0.0125 | -0.1436 | 0.1736 | 0.5983 | 0.7993 | 1.0346 | 1.3338 | 1.7150 | 2.2001 | 2.7873 |
| 7 | 0.0420 | 0.0693 | 0.1171 | 0.5162 | 1.0233 | 1.3335 | 1.7150 | 2.2007 | 2.8198 | 3.5882 |
| 8 | 0.0164 | -0.2235 | -0.4426 | 0.2778 | 1.0568 | 1.7080 | 2.1999 | 2.8198 | 3.6093 | 4.6151 |
| 9 | -0.0028 | -0.2892 | -0.9883 | 0.2583 | 1.3891 | 2.1567 | 2.7872 | 3.5882 | 4.6151 | 4.6821 |
| 10 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

*Change in delta over steps:*



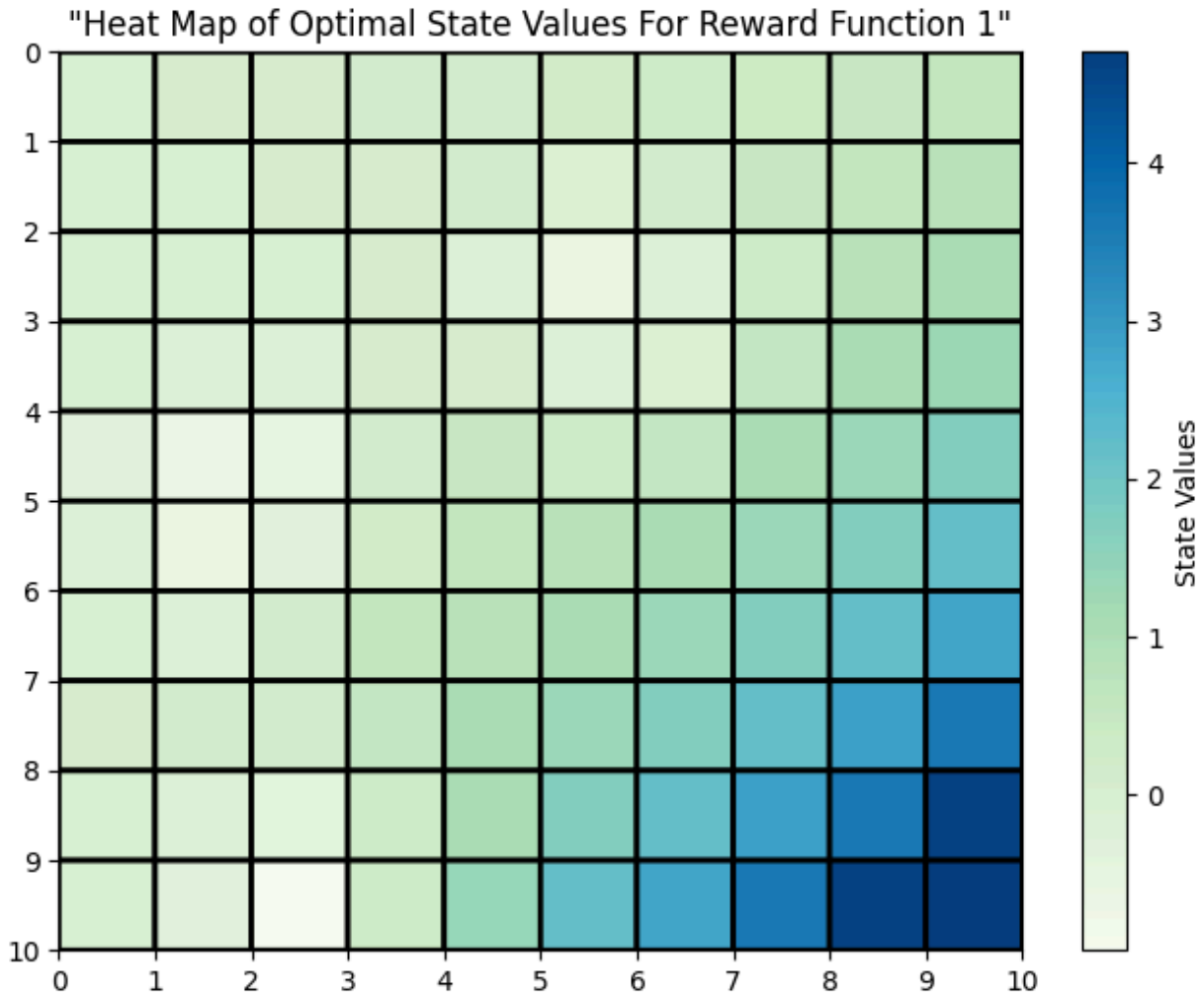

We can see that in each step, the values are increasing. As we can see in the change in delta plot, first the values increase drastically i.e the delta values are high and then it starts becoming constant by the time it reaches the last step.

State 99 has the highest optimal value which is evident because state 99 has the highest reward for reward function 1. As one moves further from state 99, the optimal value of each state decreases slowly. In addition, the blocks of states which yield negative reward show negative optimal values. The neighboring states near these blocks also have a lower optimal value compared to the neighboring states near the state yielding the highest reward. In other words, as an agent moves towards states with low reward, the optimal values of the neighboring states will progressively decrease, whereas if an agent moves towards desirable states with higher reward, the optimal values of the states it faces will progressively increase. Most of the states have a value of 0 during the earlier steps, yielding a sparse matrix. However, as steps increase, the state values near state 99 (state with highest reward) starts to increase gradually, while the state values near the states with negative reward start to decrease progressively. Since state 99 has the highest reward, the values increase much faster for state 99 and its neighboring states. As the algorithm nears convergence, the sparsity disappears, with most of the states assigned a certain non-zero optimal value, with the states near state 99 being assigned very high and positive values and states near the blocks with negative reward being assigned negative values. We can also observe that the rate of change of state values is faster during initial steps, following a logarithmic rate of convergence.

**QUESTION 3:**

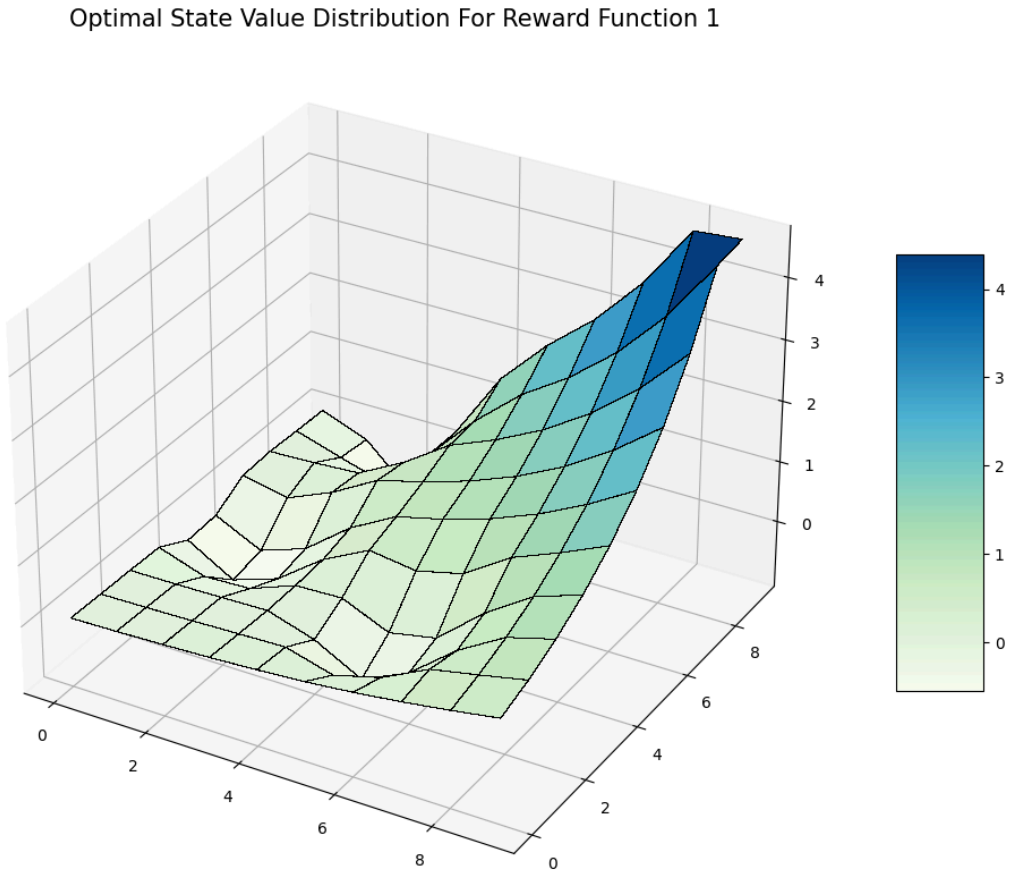
**Generate a heat map of the optimal state values across the 2- D grid. For generating the heat map, you can use the same function provided in the hint earlier (see the hint after question 1).**

The heatmap for the optimal state values for Reward Function 1 is:



**QUESTION 4:**

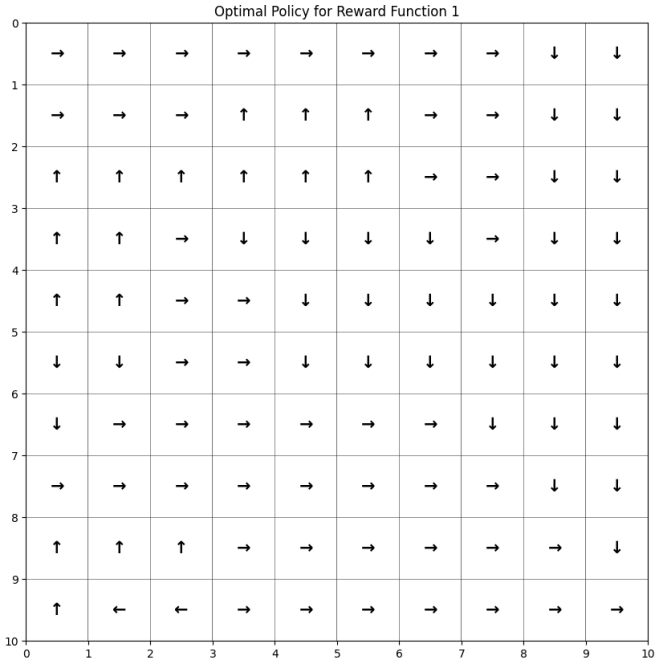
**Explain the distribution of the optimal state values across the 2-D grid. (Hint: Use the figure generated in question 3 to explain)**




We can notice from the heatmap that the value of the state 99 is the maximum. This happens because reward function 1 gives maximum reward to the state 99. The higher optimal value a state has, the higher reward it can obtain. As you move away from state 99, the optimal state decreases gradually. The optimal state increases as expected i.e it increases gradually as the agent moves towards state 99. As we go close to the exit, the value increases. When we are far from the exit, the values are low. As one moves further from state 99, which is the state providing the highest reward, the optimal value of each state decreases slowly. In addition, the neighboring states near the blocks yielding negative reward also have a lower optimal value compared to the neighboring states near the state yielding the highest reward. In other words, as an agent moves towards states with low reward, the optimal values of the neighboring states will progressively decrease, whereas if an agent moves towards desirable states with higher reward, the optimal values of the states it faces will progressively increase. There is a visible, progressive and gradual decay of optimal values surrounding the state with the highest reward. This is due to the discount factor in the Bellman equation, which discounts future rewards. The patterns seen in the heatmap of reward function 1 are roughly visible in the heatmap of optimal values. There are three blocks of negative rewards for reward function 1. One can almost make out the same three blocks from the heatmap of optimal values within the same expected regions in the state space. In addition, the region near the state with the highest reward is also brighter. This provides us with an intuition that it is possible to extract reward function by observing how the environment or an expert behaves. This is roughly what is done in inverse reinforcement learning (IRL).

**QUESTION 5:**

**Implement the computation step of the value iteration algorithm (lines 14-17) to compute the optimal policy of the agent navigating the 2-D state-space. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The optimal actions should be displayed using arrows. Does the optimal policy of the agent match your intuition? Please provide a brief explanation. Is it possible for the agent to compute the optimal action to take at each state by observing the optimal values of it's neighboring states? In this question, you should have 1 plot.**

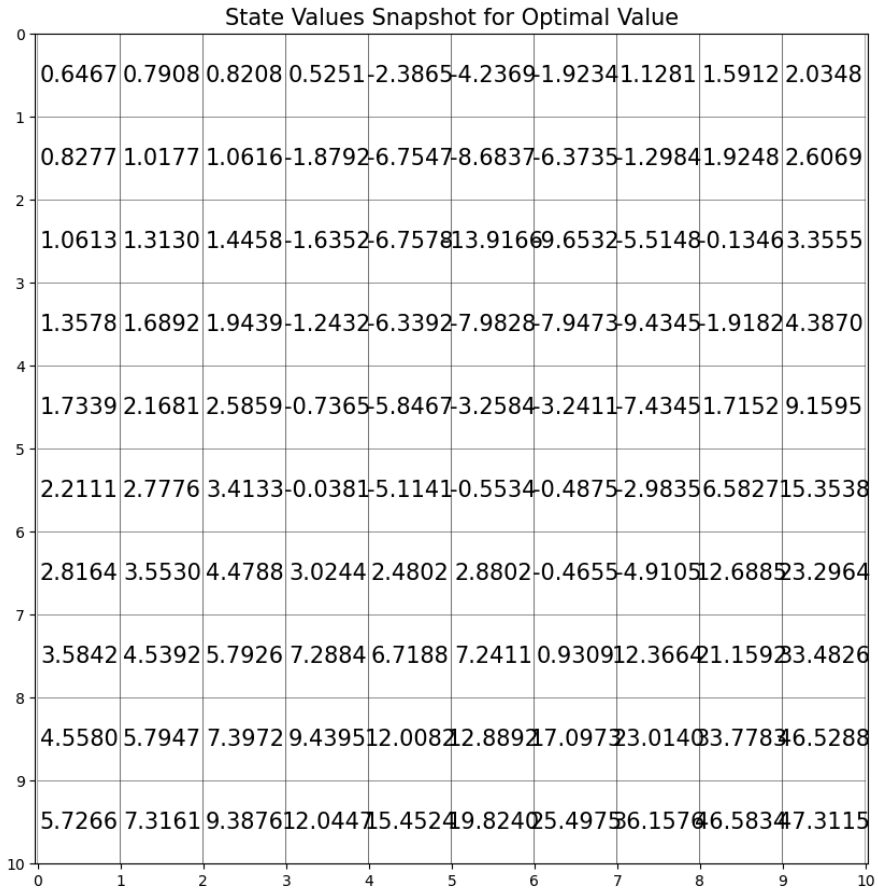



From the above plot, we can see that the arrows are all leading or converging to state 99. Intuitively, this is what was supposed to be observed as the reward for state 99 is the highest and the agent should eventually find a path to that state. This can also be verified from the optimal values and the heat map of the optimal values which we depicted in the previous questions. According to the computation step of the Value-Iteration algorithm, the optimal policy will solely depend on the transition probabilities, discount factor and the optimal value of its neighboring states. However, in this case, we are not varying gamma i.e. the discount factor. Thus, the two factors which affect the optimal policy computation are the transition probabilities and the optimal value of the neighboring states. The algorithm then takes the max of all directions by considering the optimal values and the transition probabilities of the respective neighboring states. Thus, it first considers the optimal values of the neighboring states and then chooses an optimum value for itself. It is in this manner that the entire optimum policy is built. Also, the agent is likely to move in the direction which gives the best optimal value. In other words, all the other states tend to take actions to come nearer to the last state in order to get a higher reward. That is, all states go to their neighbor states with higher values. It is possible for the agent to compute the optimal action to take at each state by observing the optimal values of its neighboring states. This is evident when we check which way the arrow is pointing given the optimal values of neighboring states. At each state, the arrow always points towards that neighboring state whose optimal value is highest among all neighbors. As a result, for this particular case, even if the agent is unaware of future optimal values, the agent can still build an optimal policy just by observing the local optimal values within the neighborhood of each state. Thus we conclude that the agent computes the optimal action at each state by observing the optimal values of its neighboring states and deciding the best among them.

**QUESTION 6:**

**Modify the environment of the agent by replacing Reward function 1 with Reward function 2. Use the optimal state-value function implemented in question 2 to compute the optimal value of each state in the grid. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal value of that state. In this question, you should have 1 plot.**

*Using the value iteration algorithm to get the optimal value of the state:*

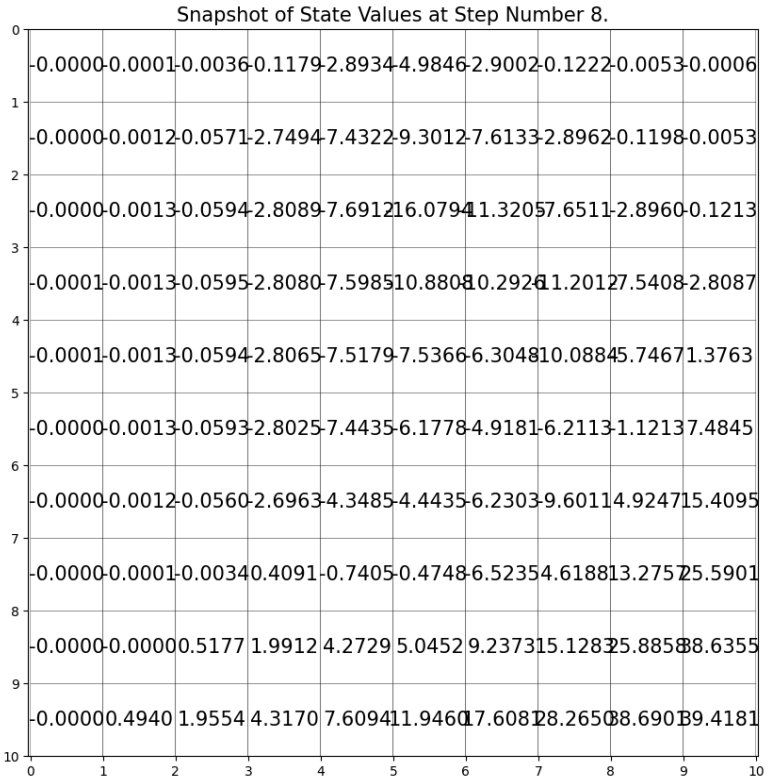



From the above plot, we can see that the values are close to 0 i.e they are low in the initial states and then they start increasing as we reach the states with the maximum reward value.

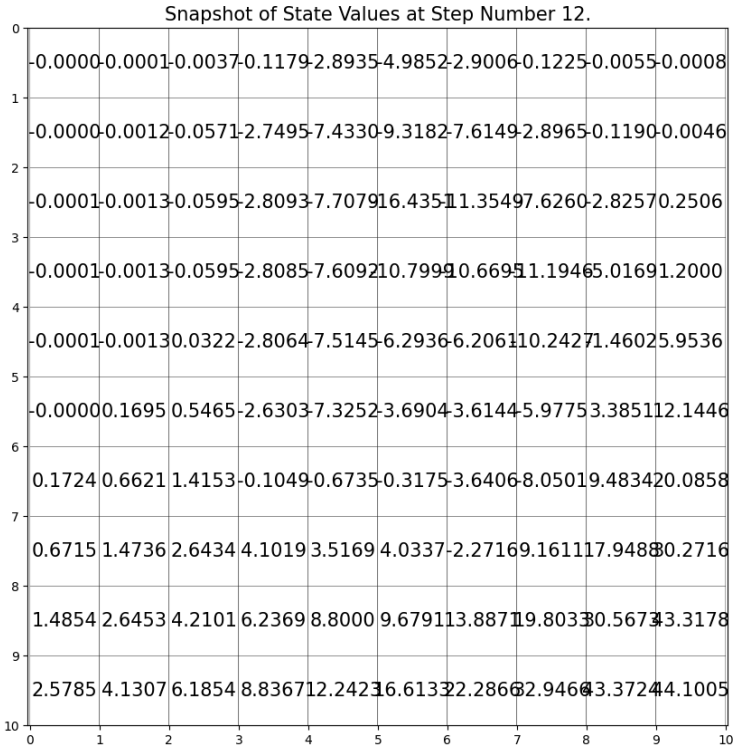
*Snapshots of state values in 8 different steps linearly distributed:*
Number of steps for Value Iteration Algorithm to converge: 32
Number of intermediate steps captured: 8
The step numbers captured are: [4, 8, 12, 16, 20, 24, 28, 32]

Step Number:  4
Value of Delta:  4.842216000000004

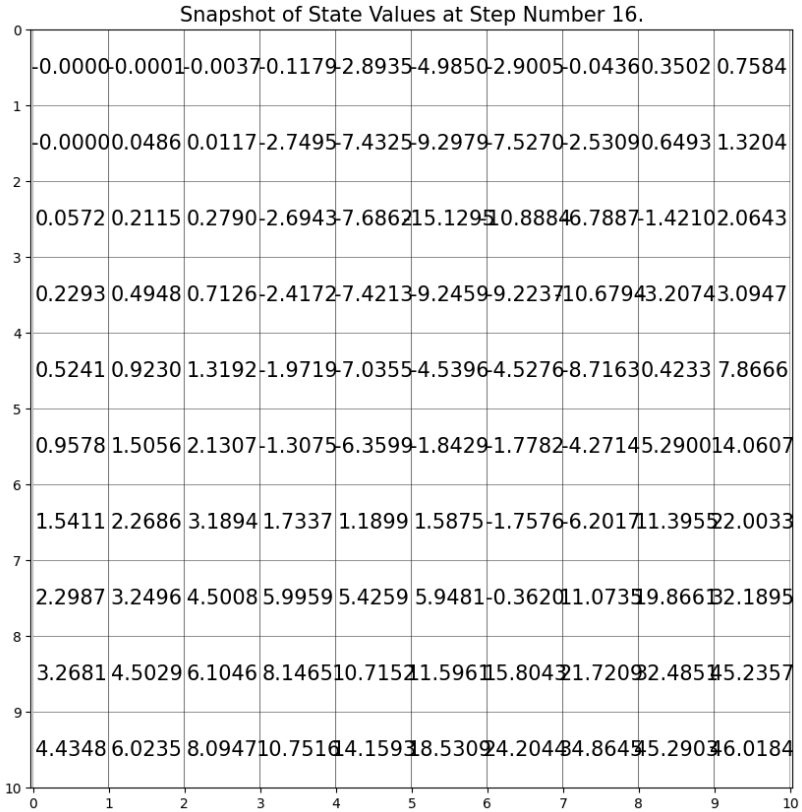

Step Number:  8
Value of Delta:  1.9827083568038404



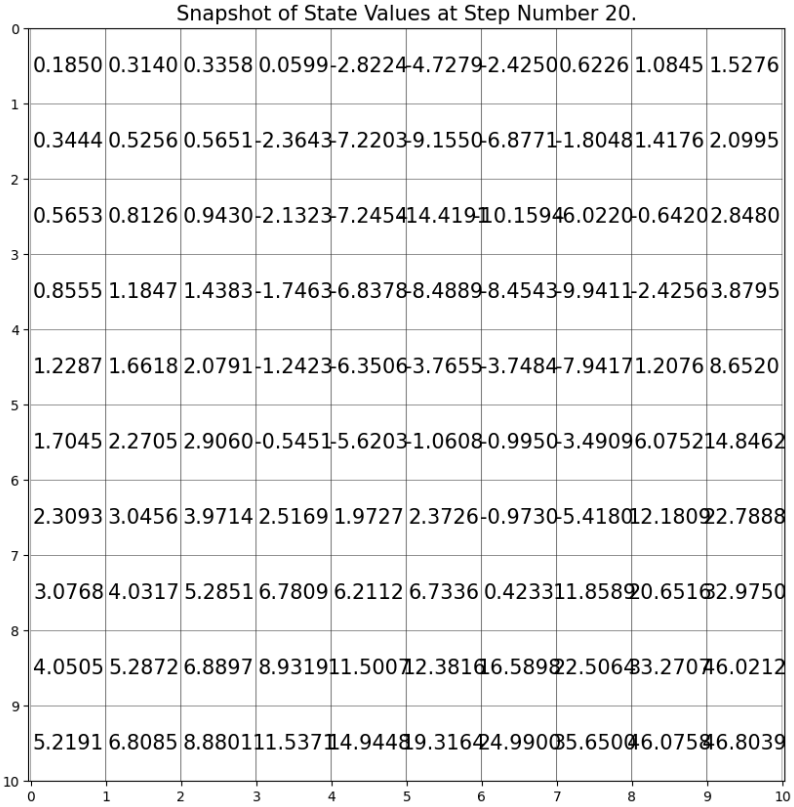
Step Number: 12
Value of Delta: 0.8121123856020347



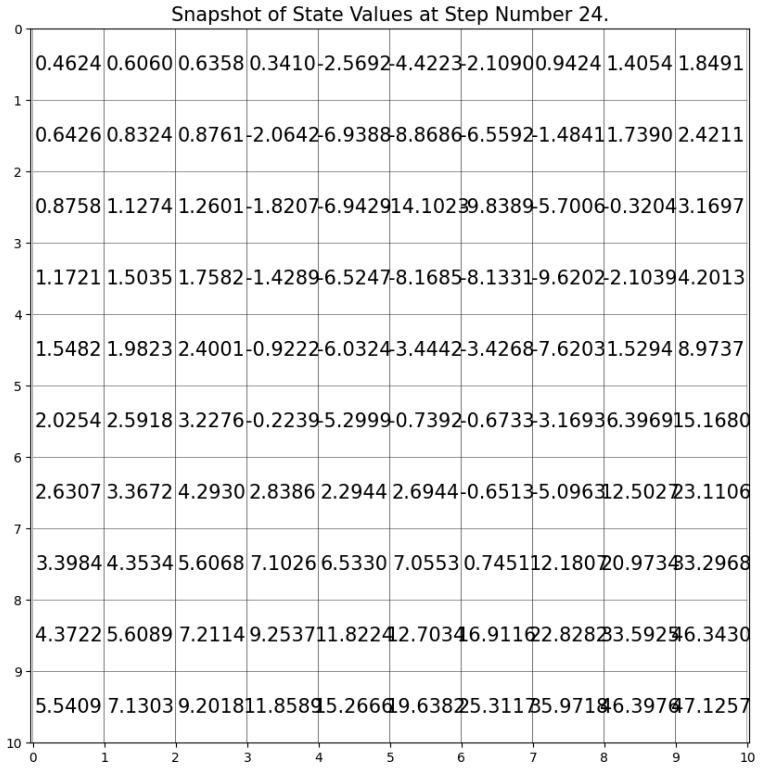

Step Number: 16
Value of Delta: 0.3326410412955383



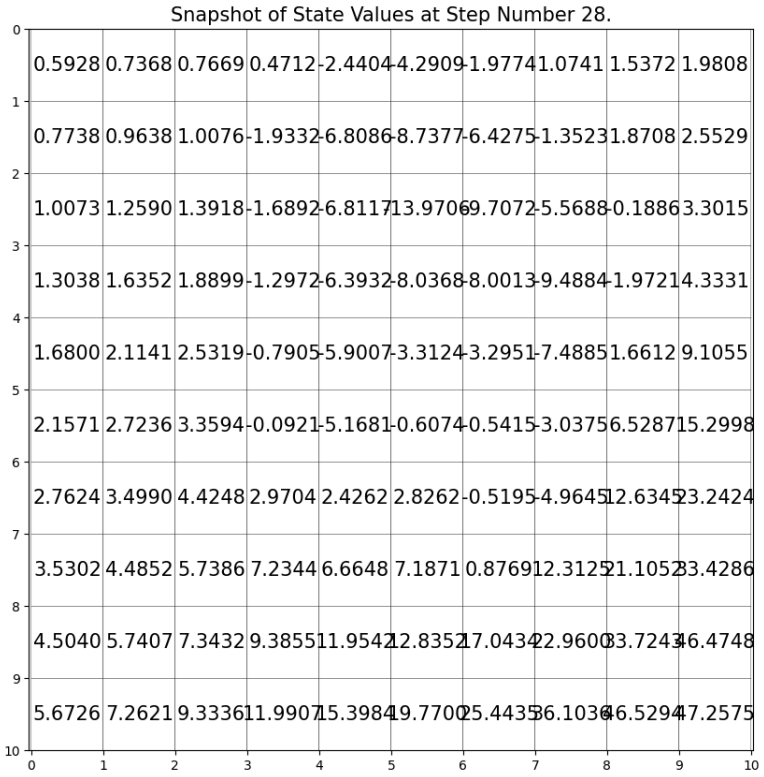
Step Number:  20
Value of Delta:  0.13624976237936437



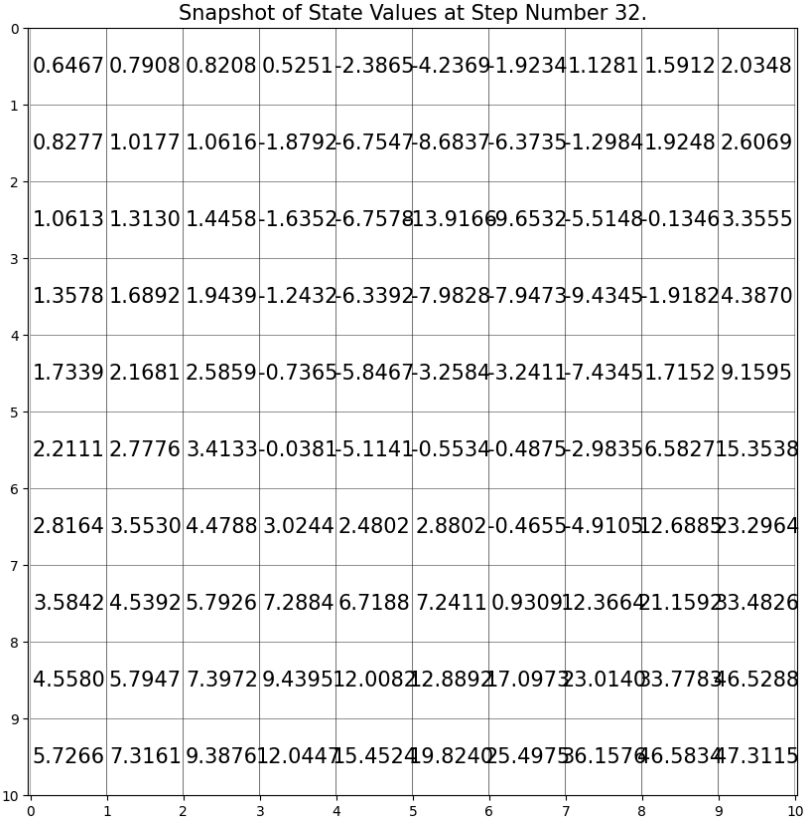

Step Number:  24
Value of Delta:  0.05580790246924039



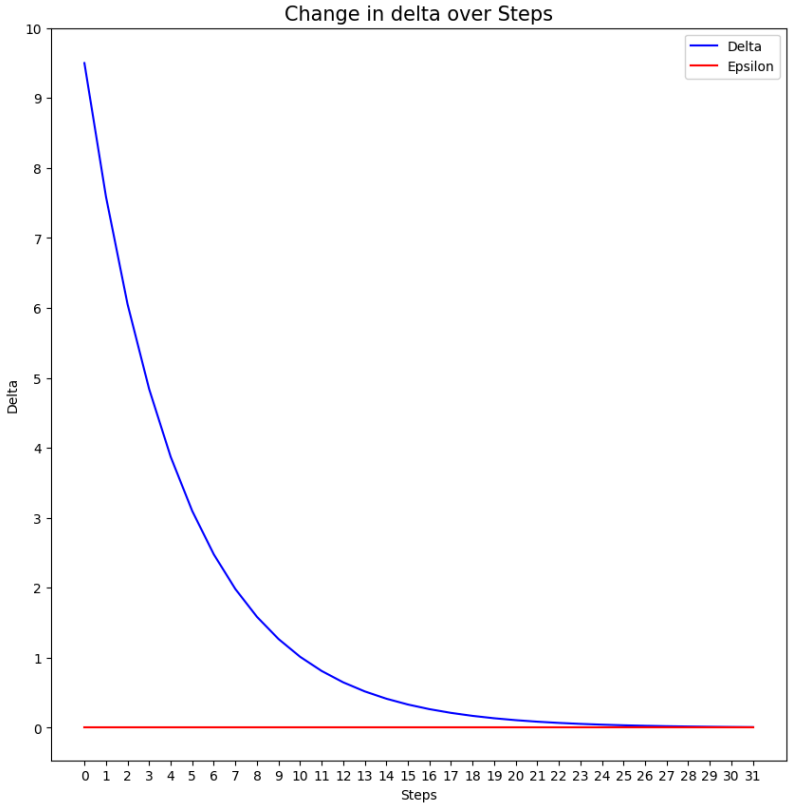
Step Number:  28
Value of Delta:  0.02285891684528707




Step Number:  32
Value of Delta:  0.00936301233963377



*Change in delta over steps:*




We can see that in each step, the values are increasing. As we can see in the change in delta plot, first the values increase drastically i.e the delta values are high and then it starts becoming constant by the time it reaches the last step.

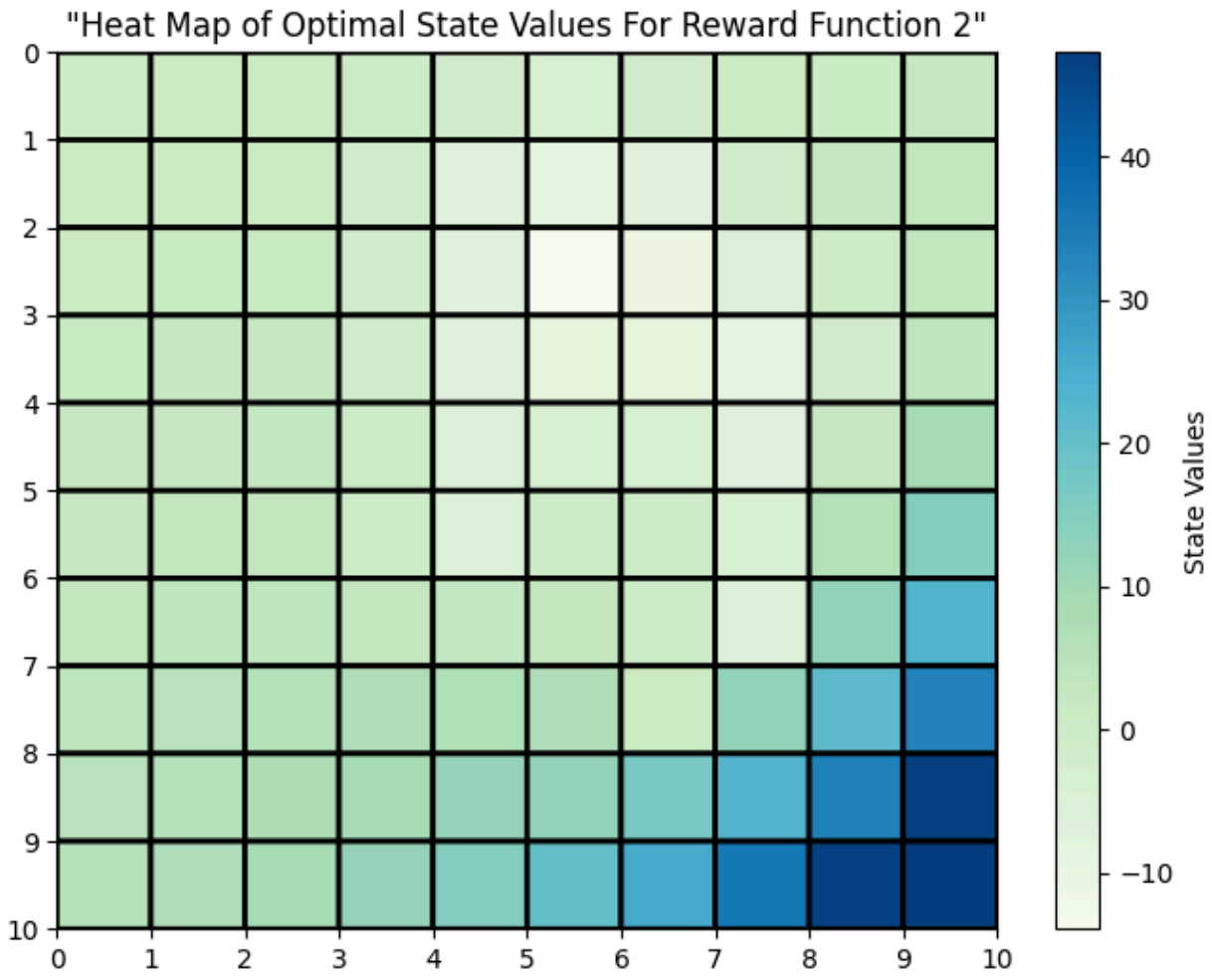
State 99 has the highest optimal value which is evident because state 99 has the highest reward for reward function 2. As one moves further from state 99, the optimal value of each state decreases slowly. In addition, the blocks of states which yield negative reward show negative optimal values. The neighboring states near these blocks also have a lower optimal value compared to the neighboring states near the state yielding the highest reward. In other words, as an agent moves towards states with low reward, the optimal values of the neighboring states will progressively decrease, whereas if an agent moves towards desirable states with higher reward, the optimal values of the states it faces will progressively increase. Most of the states have a value of 0 during the earlier steps, yielding a sparse matrix. However, as steps increase, the state values near state 99 (state with highest reward) starts to increase gradually, while the state values near the states with negative reward start to decrease progressively. Since state 99 has the highest reward, the values increase much faster for state 99 and its neighboring states. As the algorithm nears convergence, the sparsity disappears, with most of the states assigned a certain non-zero optimal value, with the states near state 99 being assigned very high and positive values and states near the blocks with negative reward being assigned negative values. We can also observe that the rate of change of state values is faster during initial steps, following a logarithmic rate of convergence.

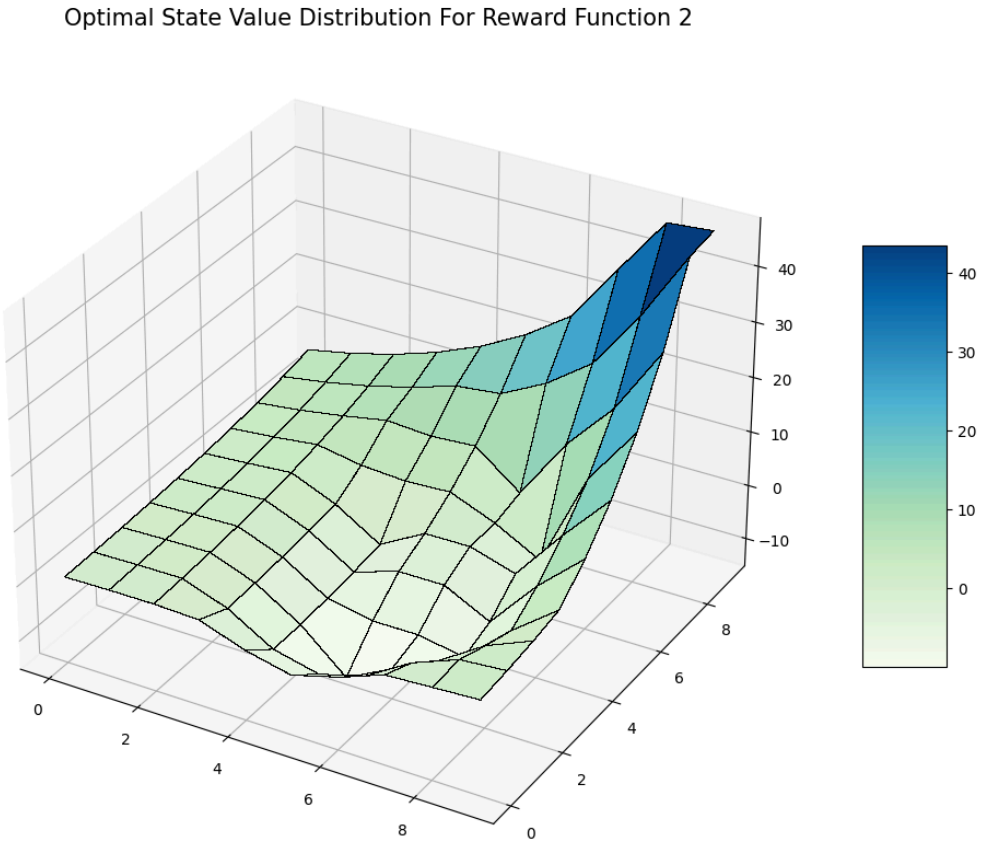
**QUESTION 7:**

**Generate a heat map of the optimal state values (found in question 6) across the 2-D grid. For generating the heat map, you can use the same function provided in the hint earlier.**

**Explain the distribution of the optimal state values across the 2-D grid. (Hint: Use the figure generated in this question to explain)**

The heatmap for the optimal state values for Reward Function 2 is:





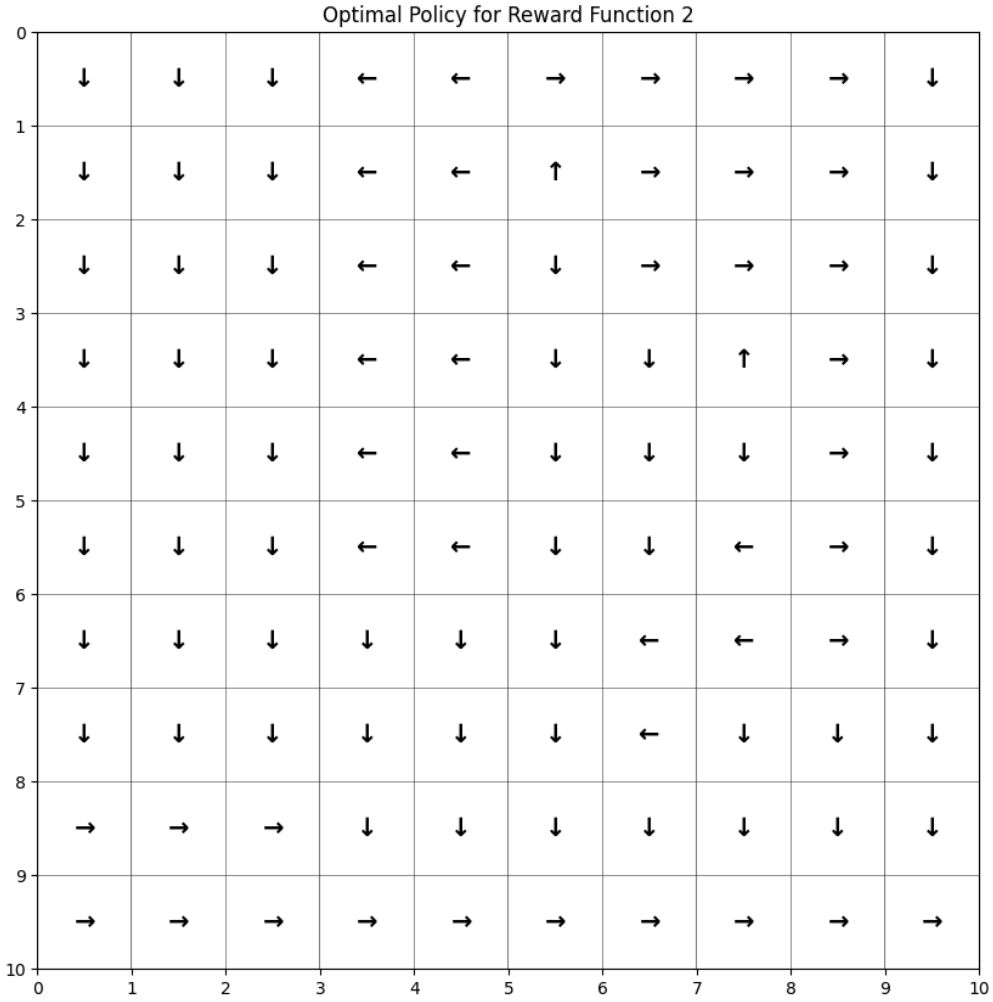

In reward function 2, we observe that some of the states have a negative reward i.e. -100. This means that if the agent moves to that state, it will be getting a negative reward, which is obviously not desirable! Hence, the agent should try to avoid these states. Essentially this means that such states should have lesser optimal state values that the states to which the agent actually wants to go. These states mostly beget a negative optimum state value. This is what is depicted in the above heat map of optimal state values. Here too, the reward for reaching the 99th state is the highest. Thus, the agent will always try to get to this state by avoiding the states as much as possible which give negative reward. Such states with negative optimal values are depicted in lighter green shades in the heat map. This is very prominently observed in the 52nd state of the grid which begets the lightest color. The lightest color signifies that it has the least optimal value. This can be justified because this state is surrounded by states on three sides (up: 51st state, left: 42nd state, right: 62nd state) which have -100 reward. That is why the optimal value of this state is very low, because it is highly possible that if the agent comes to this state then it will most likely land up in one of the neighboring states which have -100 reward. Hence, the agent should avoid this state. That is why it is justified for this state to have a very low optimal state value. Likewise, for all other states which have light color in the above heat map, their optimal state values are also low as compared to the states where the agent will actually benefit. Other states which have also been shown in the shades of green color (but a lighter shade) signify that the optimal values are close to 0 but positive. This is starkly observed in the upper left part of the grid. If the agent lands up in some state in the bottom right part of the grid, then it will be easier for him to go to the 99th state which has the highest reward. Therefore, these states in the bottom right part of the grid are signified by a high optimal value and a darker blue shade which essentially means that it is beneficial for the agent to visit these states. Such is the distribution of the optimal state values across the 2-D grid.
As one moves further from state 99, which is the state providing the highest reward, the optimal value of each state decreases slowly. In addition, the neighboring states near those states yielding negative reward also have

a lower optimal value compared to the neighboring states near the state yielding the highest reward. In other words, as an agent moves towards states with low reward, the optimal values of the neighboring states will progressively decrease, whereas if an agent moves towards desirable states with higher reward, the optimal values of the states it faces will progressively increase. The further a state is from the state yielding the highest reward, the lower its value. The patterns seen in the heatmap of reward function 2 are roughly visible in the heatmap of optimal values. We see that the negative rewards form a snake-like chain in the state-space. This chain is roughly visible in the heatmap, with the light green regions corresponding to the states with lowest rewards. In addition, the region near the state with the highest reward is also darker. This provides us with an intuition that it is possible to extract reward function by observing how the environment or an expert behaves. This is roughly what is done in IRL.

We see that state 52 has the lowest optimal value and is being shown the lightest in the heatmap. This is because state 52 is surrounded by states with negative rewards on three sides with only one path that does not penalize the agent. Thus, value iteration encourages the agent to avoid this state, as the probability of landing in a state with negative reward is the highest among all the states if the agent lands on this state. There is a visible, progressive and gradual decay of optimal values surrounding the state with the highest reward. This is due to the discount factor in the Bellman equation, which discounts future rewards.

**QUESTION 8:**

**Implement the computation step of the value iteration algorithm (lines 14-17) to compute the optimal policy of the agent navigating the 2-D state-space. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The optimal actions should be displayed using arrows. Does the optimal policy of the agent match your intuition? Please provide a brief explanation. In this question, you should have 1 plot.**



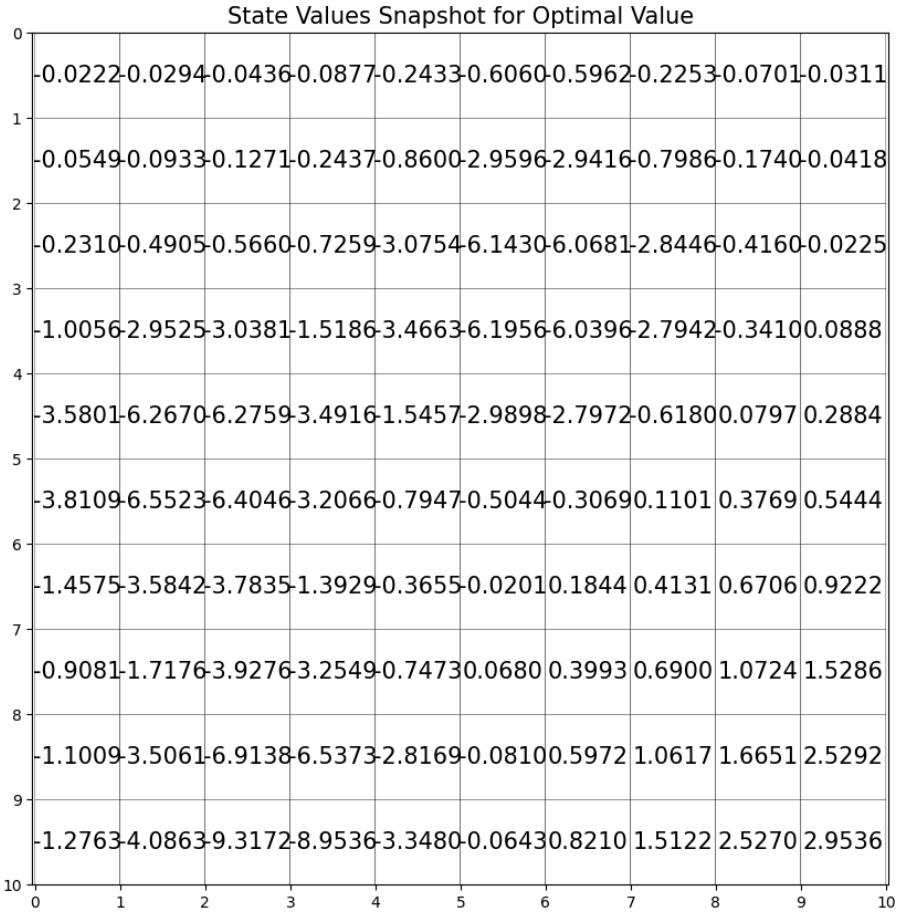

In this part of the project, since the reward function has some negative value, it will have some role to play in the estimation and computation step. On the basis of previous results and also intuition, the agent will try to avoid the states which give a negative reward. This is depicted in the grid above where we can see that the arrows are diverging from the state which give a negative reward. For example, if we consider the same example of the 52nd state, then we can see that the arrow in the 52nd state is pointing down, i.e. towards the 53rd state. This is correct because the remaining 3 neighboring states have a negative reward which should be avoided. These 3 states also point in a direction away from the center i.e. the 52nd state. This is desirable, because if not there would have been a chance that the agent would have got locked in the 52nd state. In other states surrounding the states with reward -100, we observe that the arrows do not point to these negatively rewarding states. Again, this is desirable, because we do not want the agent to go to these states. We also observe in the bottom right part of the grid that almost all the arrows are leading to the 99th state. The agent will only move to the state which has the maximum optimum value among all the neighboring states. Again, all this is intuitive as well as justified by our optimal values obtained previously. Thus, the actions of the agent as shown in this grid is validated and justified by the optimal values and the heat map obtained previously.

**QUESTION 9:**

**Change the hyper parameter w to 0.6 and find the optimal policy map similar to previous question for reward functions. Explain the differences you observe. What do you think about value of new w compared to previous value? Choose the w that you think give rise to better optimal policy and use that w for the next stages of the project.**

***For Reward Function 1:***

*Optimal State Value:*



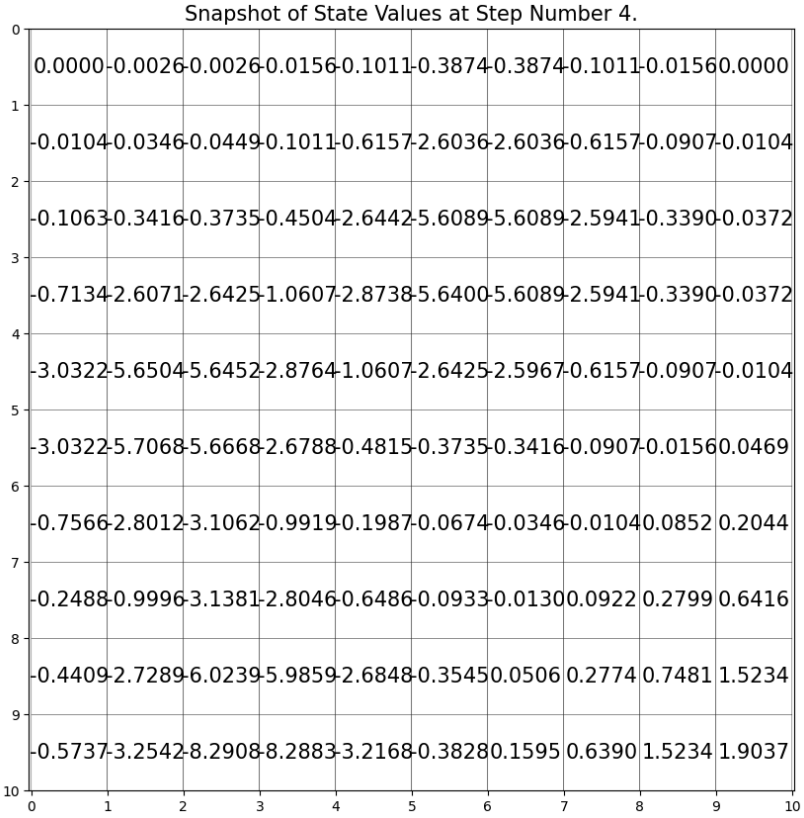

*Change in Steps:*

Number of steps for Value Iteration Algorithm to converge: 19

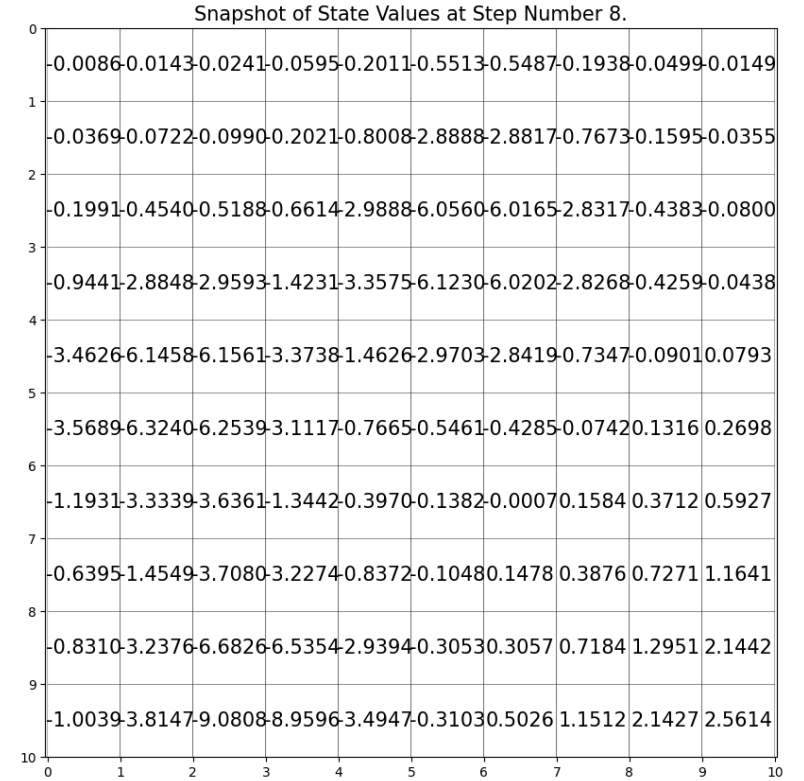
Number of intermediate steps captured: 4

The step numbers captured are: [4, 8, 12, 16]

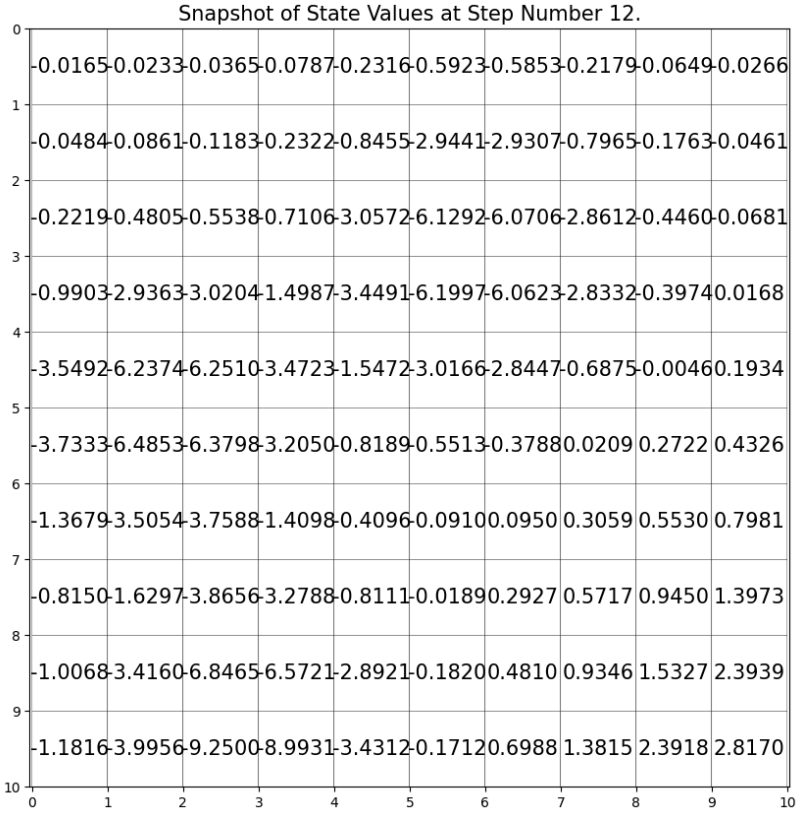
Step Number:  4
Value of Delta:  0.5820480000000003



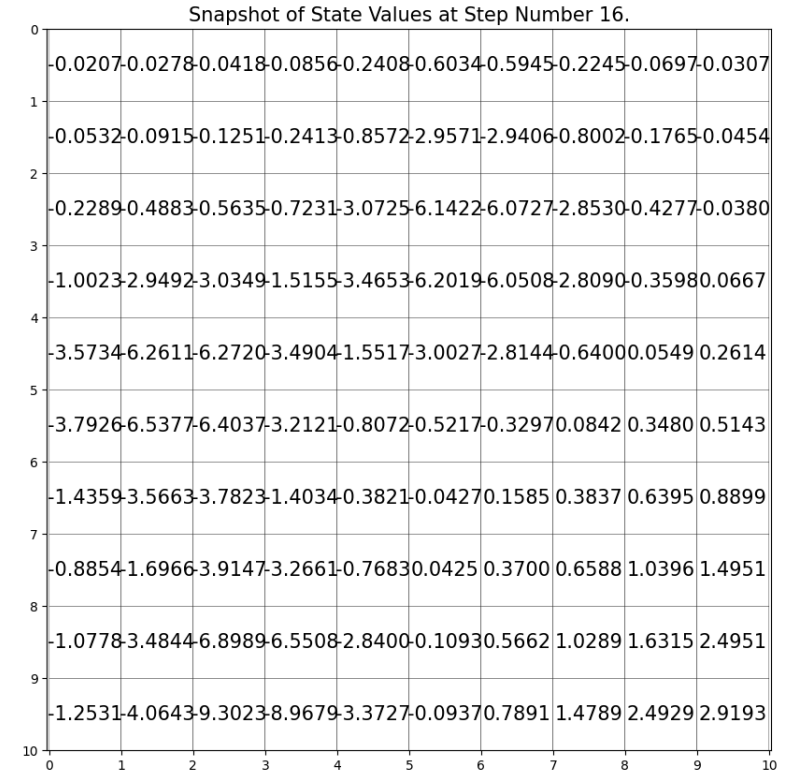

Step Number:  8
Value of Delta:  0.11072197782732784



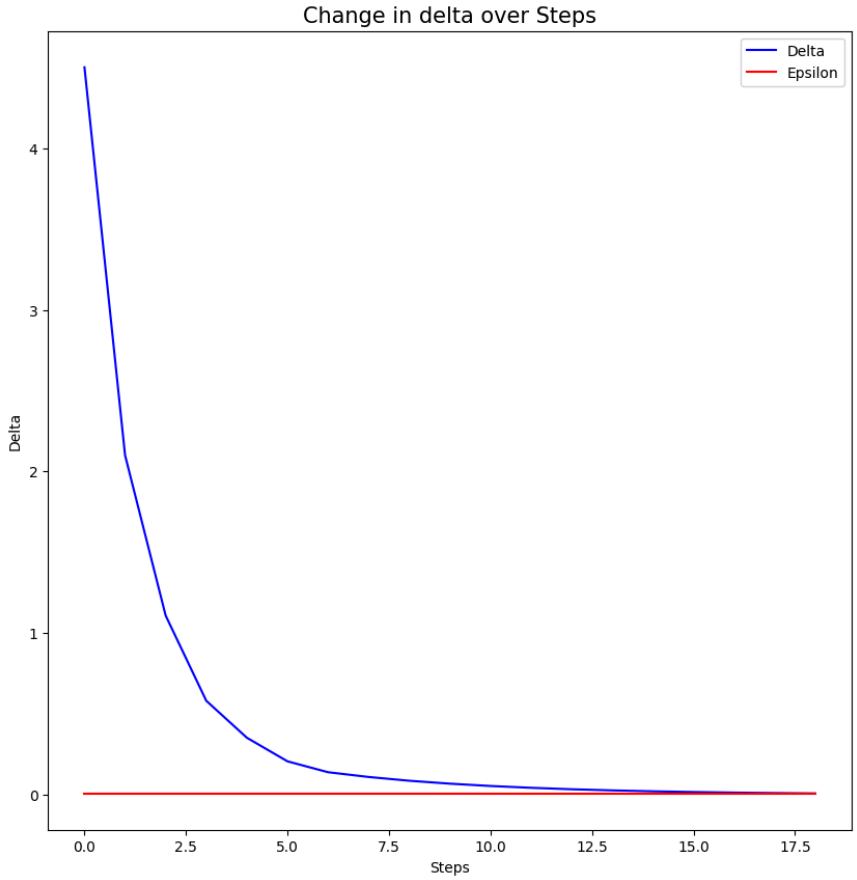
Step Number:  12
Value of Delta:  0.0437785231222545



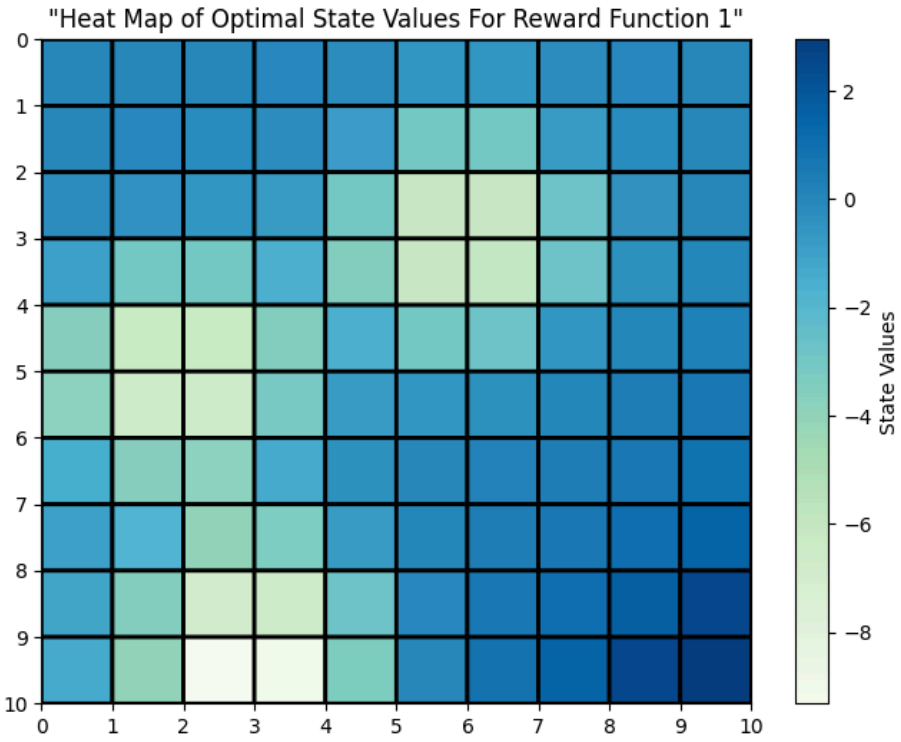

Step Number:  16
Value of Delta:  0.0176390444782073



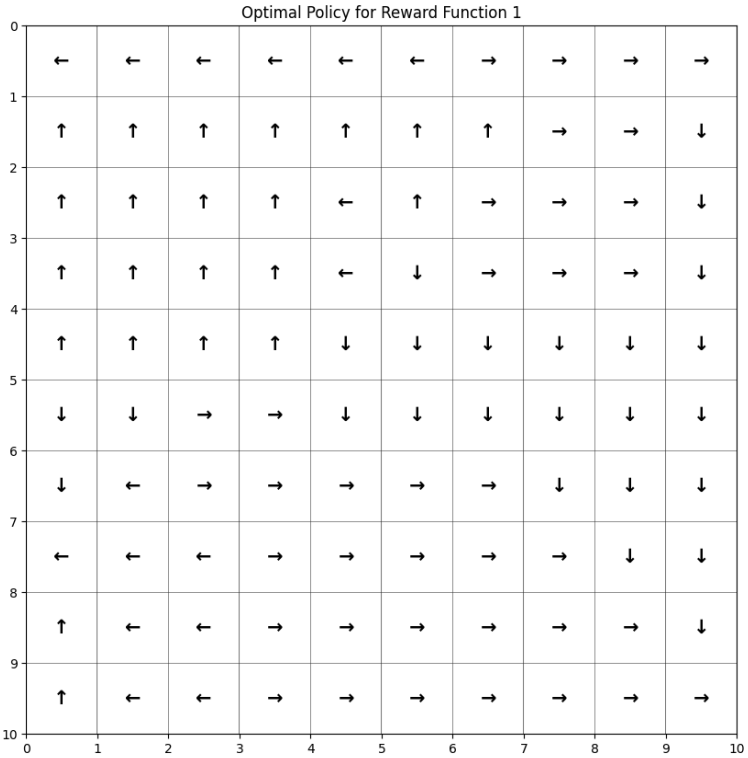
*Change in Delta over Steps:*




*HeatMap of Optimal State Values:*

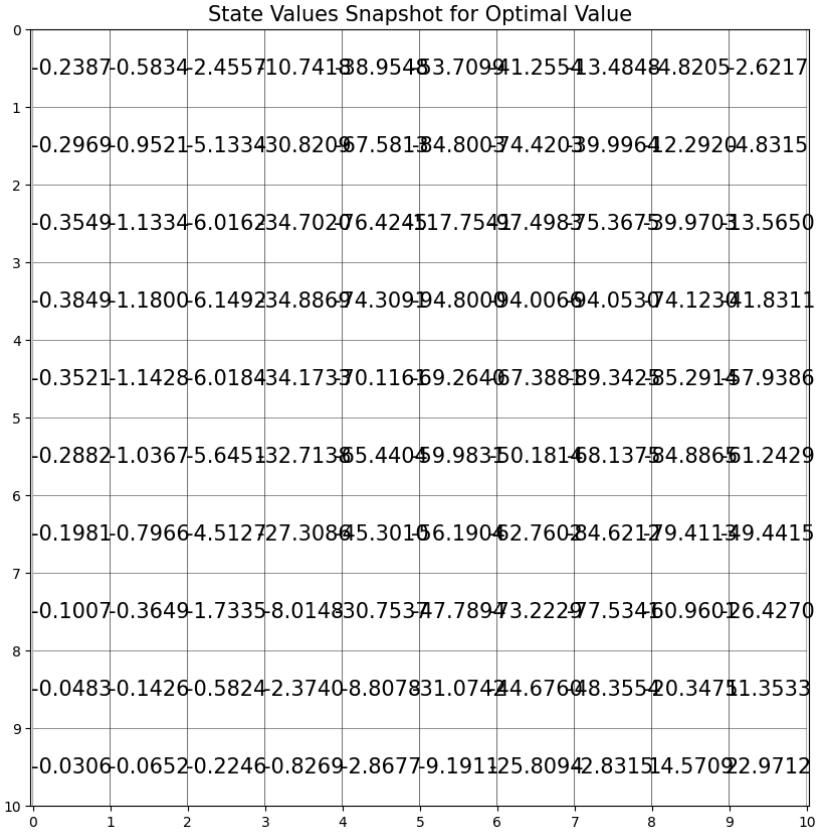


*Optimal Policy:*




***For Reward Function 2:***

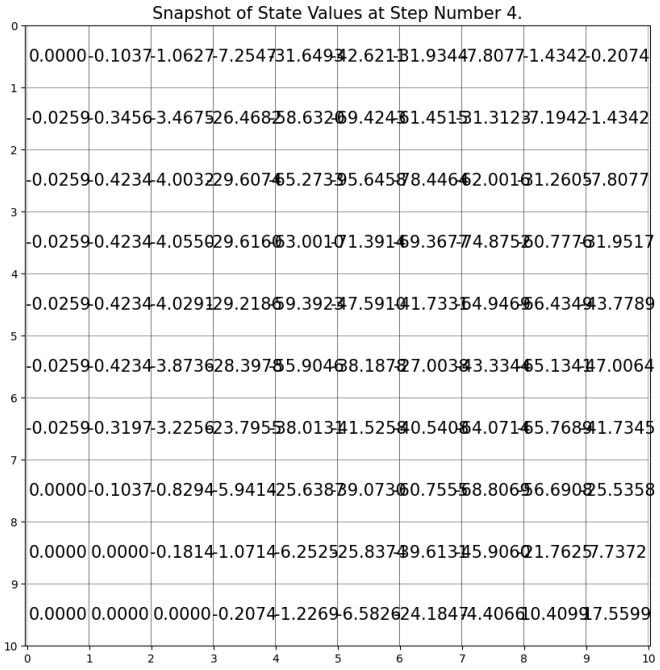
*Optimal State Value:*



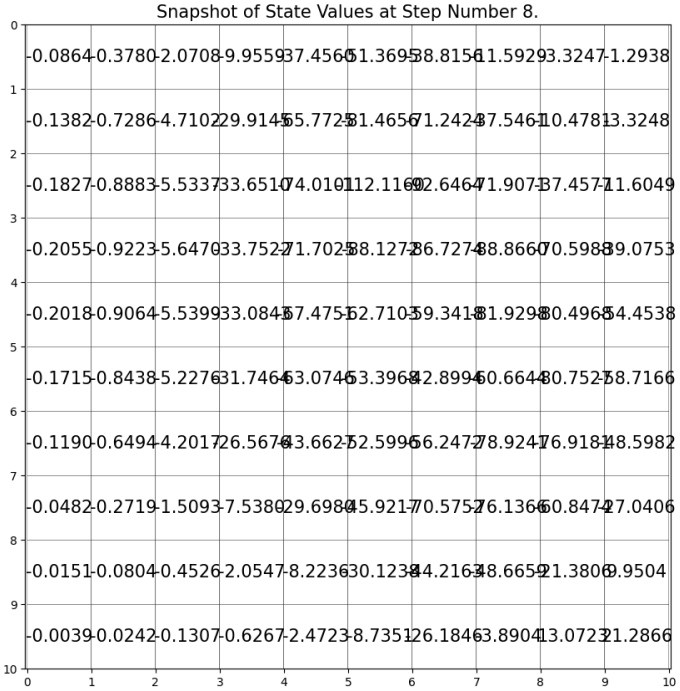
*Change in Steps:*

Number of steps for Value Iteration Algorithm to converge: 27
Number of intermediate steps captured: 6
The step numbers captured are: [4, 8, 12, 16, 20, 24]

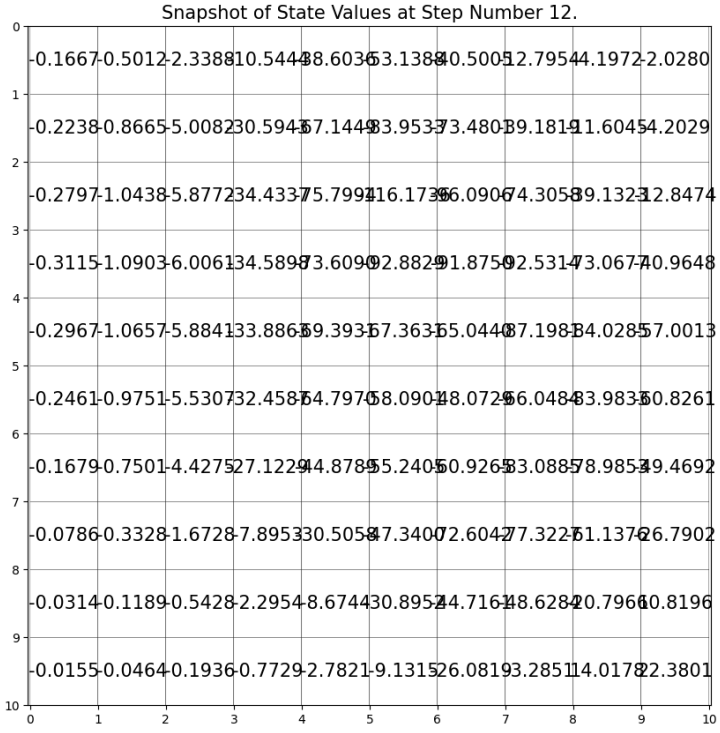
Step Number:  4
Value of Delta:  10.493760000000009



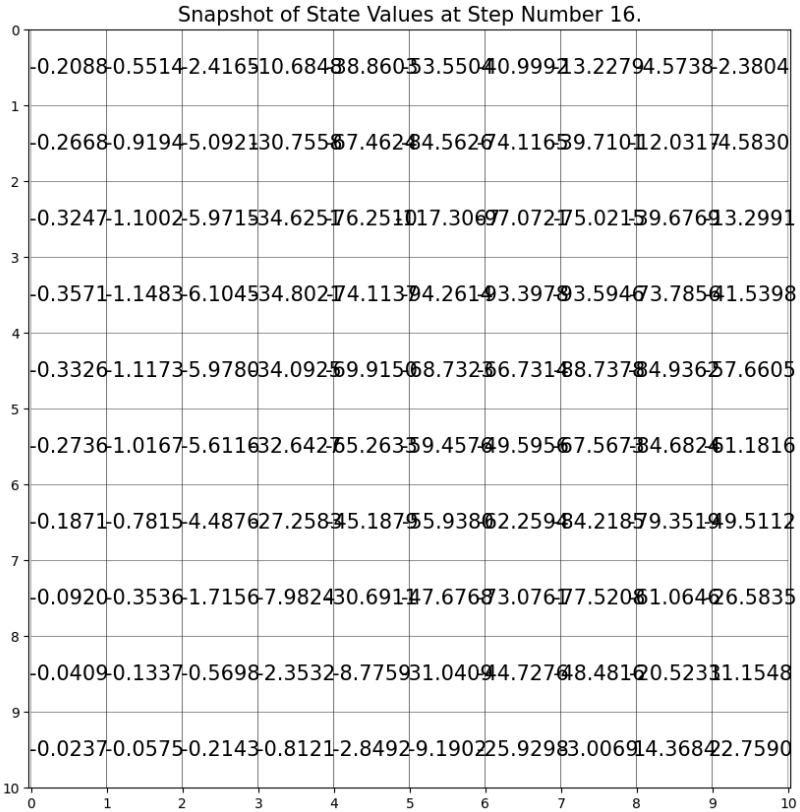

Step Number:  8
Value of Delta:  2.9731592537702483



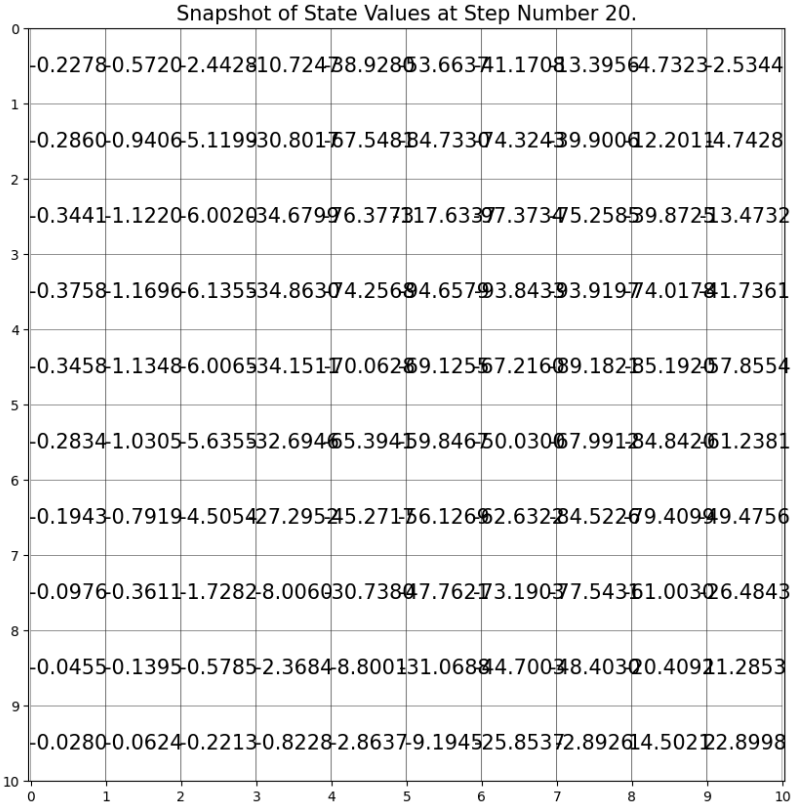
Step Number:  12
Value of Delta:  0.8741292340200459



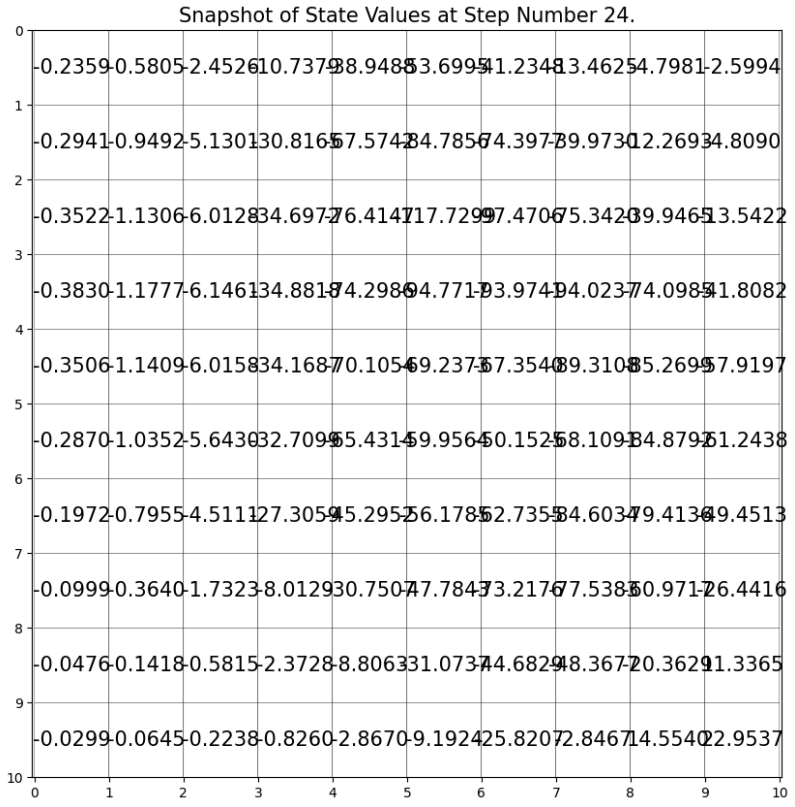

Step Number:  16
Value of Delta:  0.24937028244634973



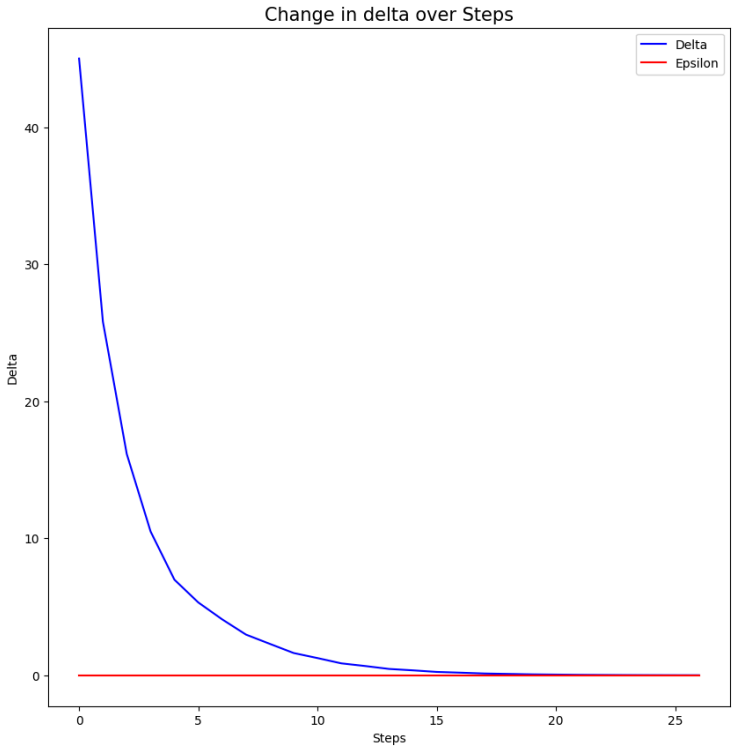
Step Number:  20
Value of Delta:  0.07201548567792315



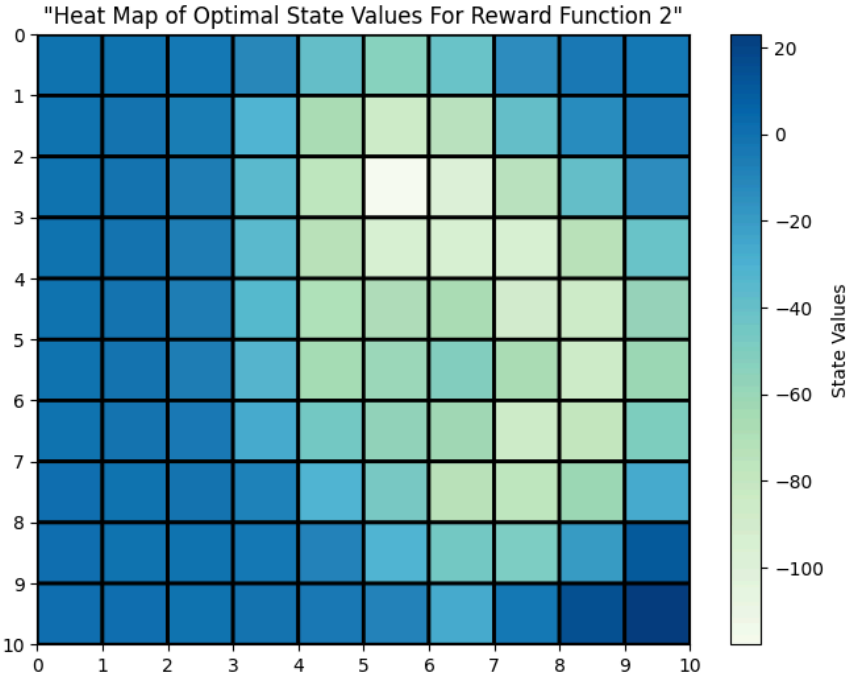

Step Number:  24
Value of Delta:  0.021060829162323103



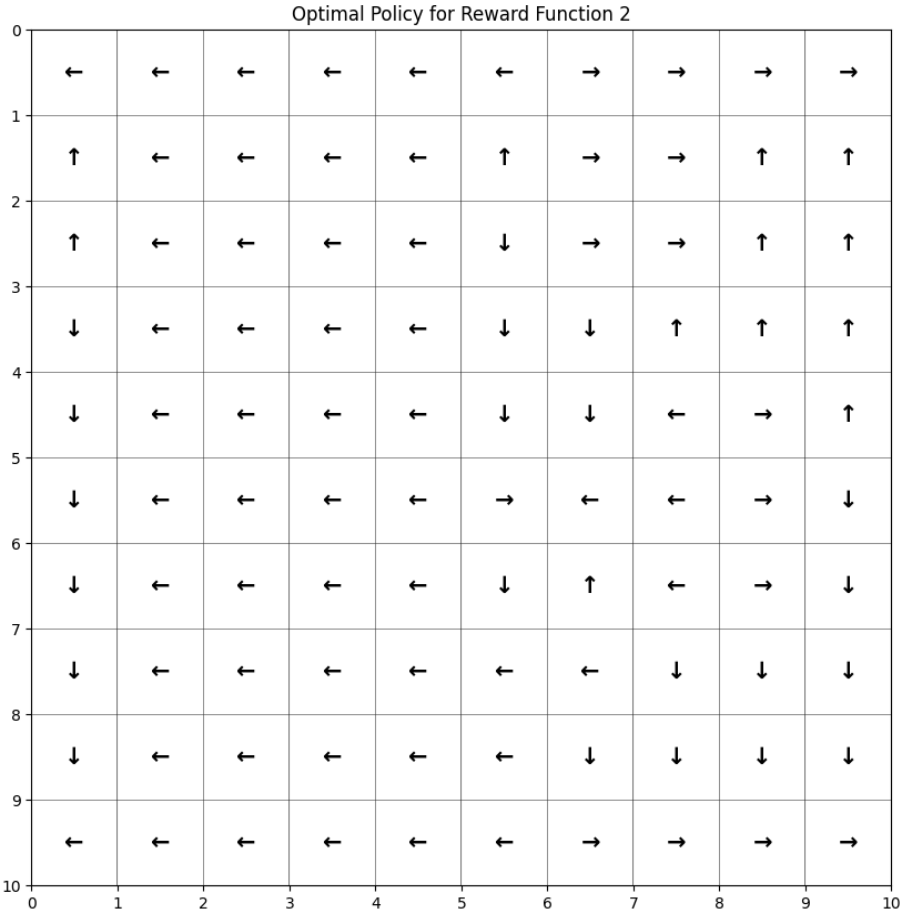
*Change in Delta over Steps:*




*HeatMap of Optimal State Values:*



*Optimal Policy:*




The optimal values for the states neighboring state 99 (state with highest reward), as well as the value for state 99 itself has decreased. The negative value of those states with negative rewards have been amplified. In summary, more states take on higher or lower values instead of just the neighboring states with positive and negative rewards, giving the agent more choice on where to move, ultimately decreasing the probability of landing in the state with highest reward.

For reward function 2, the heatmap and value plots point out that the path to the state with highest optimal value has been cut-off due to the increase in adversarial influence by the negative-reward chain over a longer state-space. This is due to amplification of negative rewards and attenuation of positive rewards.

From the policy plots, it is evident that the increase in influence of negative rewards leads to the agent often moving out of the grid. In addition, fewer paths now exist for both reward functions that can lead to the agent to state 99. This happened because the states neighboring state 99 do not provide any positive or negative reward to the agent (0 reward), causing the values for those states to be overpowered by the states where reward is negative.

We can observe some local optima in the policy plots, where the agent just oscillates between two states. This is called the deadlock condition.

All of the above reasons can be attributed to the exploration versus exploitation dilemma. We can think of it as the probability of exploration. Exploitation is making the best possible decision (by maximizing future reward), and exploration involves taking an immediately suboptimal action to gather information. While the suboptimal action will necessarily involve a reduced amount of reward in the immediate future, it may allow the agent to learn better strategies that enable policy improvement in the long term. If there is no balance between

exploration and exploitation, it would lead to a bad sample complexity. In this case, since is very high, the balance has not been achieved, leading to the agent exploring more suboptimal actions at each state rather than exploiting knowledge of optimal values reasonably. This leads to the extracted policy being suboptimal. The appropriate value of w is 0.1, as the agent is more likely to reach the state with the highest reward while having the option to perform limited exploration and less likely to get stuck in local optima or get blown off the grid. In addition, the radius of adversarial influence of negative reward chains/blocks are reduced and more paths point towards state 99 when w is 0.1.

**QUESTION 10:**

**Express c, x, D, b in terms of R, Pa, Pa1 , ti, u, λ and Rmax**

IRL is the task of extracting the reward function from the optimal policy, particularly when the reward function is not known a priori. The goal of IRL is to find the set of possible reward functions such that is an optimal policy, given state space, action space, optimal deterministic policy and state transition matrix. In IRL, the agent observes an expert's behavior and learn's the expert's reward function to generate the optimal policy. An expert's optimal policy can be used to extract the reward function, thereby computing the optimal policy of the agent. We prefer learning the reward function rather than the policy directly because the reward function provides a robust, succinct and transferable definition of the task at hand. The LP formulation of IRL is given by:

$$\max_{\mathbf{R},t_i,u_i} \text{s.t.} \begin{cases} \sum_{i=1}^{|S|}(t_i - \lambda u_i) \\ [(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R}] \geq t_i, \forall a \in A \setminus a_1, \forall i \\ (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R} \succeq 0, \ \ \forall a \in A \setminus a_1 \\ -\mathbf{u} \preceq \mathbf{R} \preceq \mathbf{u} \\ |\mathbf{R}_i| \leq R_{\max}, i = 1, 2, ....., |S| \end{cases}$$

R is the reward vector , A and S are action and state spaces respectively, Pa corresponds to transition matrix for action a, λ is the adjustable penalty coefficient, ti and ui are extra optimization variables
needed to pose the problem as LP formulation, $\gamma$ is the discount factor, Rmax is the maximum value of ground truth reward. The goal is to find a feasible value of R, also called linear feasibility problem.
For simplicity, we want to recast the LP formulation as follows:

$$\max_{\mathbf{x}} \text{s.t.} \begin{cases} \mathbf{c}^T\mathbf{x} \\ \mathbf{D}\mathbf{x} \preceq \mathbf{b}, \forall a \in A \setminus a_1 \end{cases}$$

Here, we consider the equation of the form Dx≤b. The identity matrix has been denoted by I and the dimensions of each matrix has been mentioned as their subscript. After recasting, we get the below matrix:

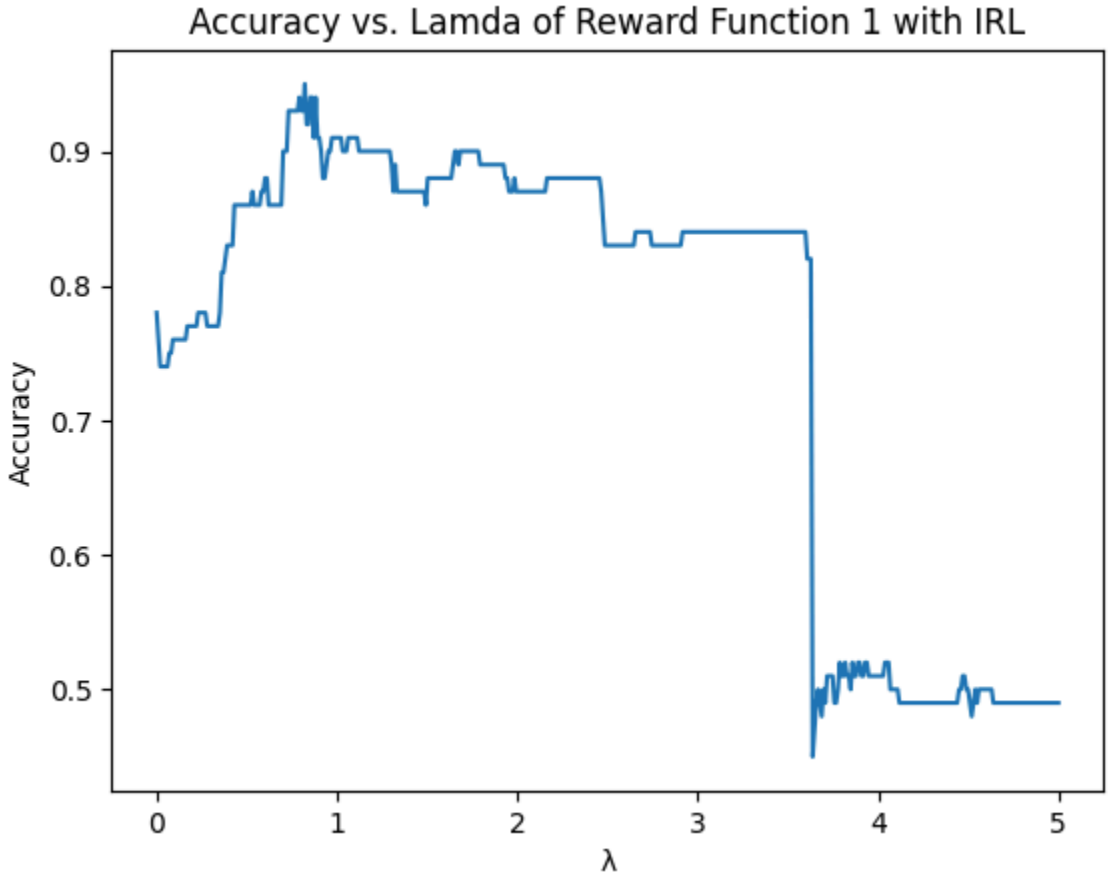
In our case |A|=4 and |S|=100.

$$c = \begin{bmatrix} \mathbf{1}_{|s|\times 1} \\ -\lambda\mathbf{1}_{|s|\times 1} \\ \mathbf{0}_{|s|\times 1} \end{bmatrix} x = \begin{bmatrix} t_{|s|\times 1} \\ u_{|s|\times 1} \\ R_{|s|\times 1} \end{bmatrix} b = \begin{bmatrix} \mathbf{0}_{(|A|-1)\cdot|s|\times 1} \\ \mathbf{0}_{(|A|-1)\cdot|s|\times 1} \\ \mathbf{0}_{|s|\times 1} \\ \mathbf{0}_{|s|\times 1} \\ R_{max}\mathbf{1}_{|s|\times 1} \\ R_{max}\mathbf{1}_{|s|\times 1} \end{bmatrix}$$

$$D = \begin{bmatrix} \mathbf{I}_{(|A|-1)\cdot|s|\times|s|} & \mathbf{0}_{(|A|-1)\cdot|s|\times|s|} & [(P_{a_{2\cdots|A|}} - P_{a_1})(I - \gamma P_{a_1})^{-1}]_{(|A|-1)\cdot|s|\times|s|} \\ \mathbf{0}_{(|A|-1)\cdot|s|\times|s|} & \mathbf{0}_{(|A|-1)\cdot|s|\times|s|} & [(P_{a_{2\cdots|A|}} - P_{a_1})(I - \gamma P_{a_1})^{-1}]_{(|A|-1)\cdot|s|\times|s|} \\ \mathbf{0}_{|s|\times|s|} & -\mathbf{I}_{|s|\times|s|} & \mathbf{I}_{|s|\times|s|} \\ \mathbf{0}_{|s|\times|s|} & -\mathbf{I}_{|s|\times|s|} & -\mathbf{I}_{|s|\times|s|} \\ \mathbf{0}_{|s|\times|s|} & \mathbf{0}_{|s|\times|s|} & \mathbf{I}_{|s|\times|s|} \\ \mathbf{0}_{|s|\times|s|} & \mathbf{0}_{|s|\times|s|} & -\mathbf{I}_{|s|\times|s|} \end{bmatrix}$$

**QUESTION 11:**

**Sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ compute OA(s) by following the process described above. For this problem, use the optimal policy of the agent found in question 5 to fill in the OE(s) values. Then use equation 3 to compute the accuracy of the IRL algorithm for this value of λ. You need to repeat the above process for all 500 values of λ to get 500 data points. Plot λ (x-axis) against Accuracy (y-axis). In this question, you should have 1 plot.**



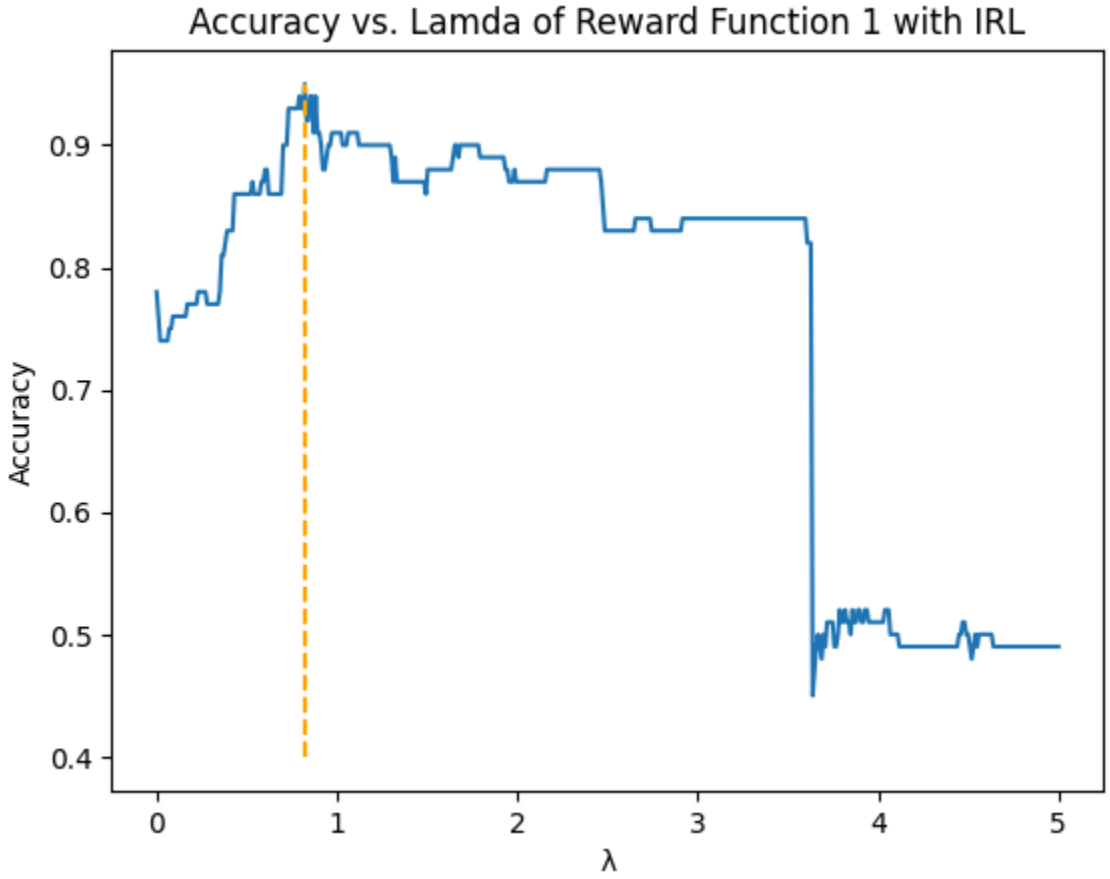

λ needs to be tuned to achieve maximum accuracy. For this, we varied lambda from 0 to 5 in 0.01 steps, getting 500 λ values. For each of these values we calculated accuracy which can be seen in the plot above. We use solvers.lp() function from cvxopt package in Python to solve the recasted LP formulation after extracting c, x D and b using the derived block matrix notation. We then perform value iteration on the extracted reward to obtain the agent's optimal policy.

We observe that accuracy initially rises as increases, reaching a globally optimal policy before dropping. Subsequently, accuracy increases slightly but then gets stuck in a local optimum. This behavior is expected because acts as a regularizer (like L1 or Lasso normalization), which promotes simpler reward vectors. Finite values of improve the robustness, conciseness, and transferability of the extracted reward function, enhancing its generalization when extracting the optimal policy. This explains the initial rise in accuracy. However, large values of result in reward vectors that underfit the tasks and lack the expressiveness needed to derive the optimal global policy, increasing the risk of getting stuck in a local optimum.

**QUESTION 12:**

**Use the plot in question 11 to compute the value of λ for which accuracy is maximum. For future reference we will denote this value as λ(1) . Please report λ(1) max**

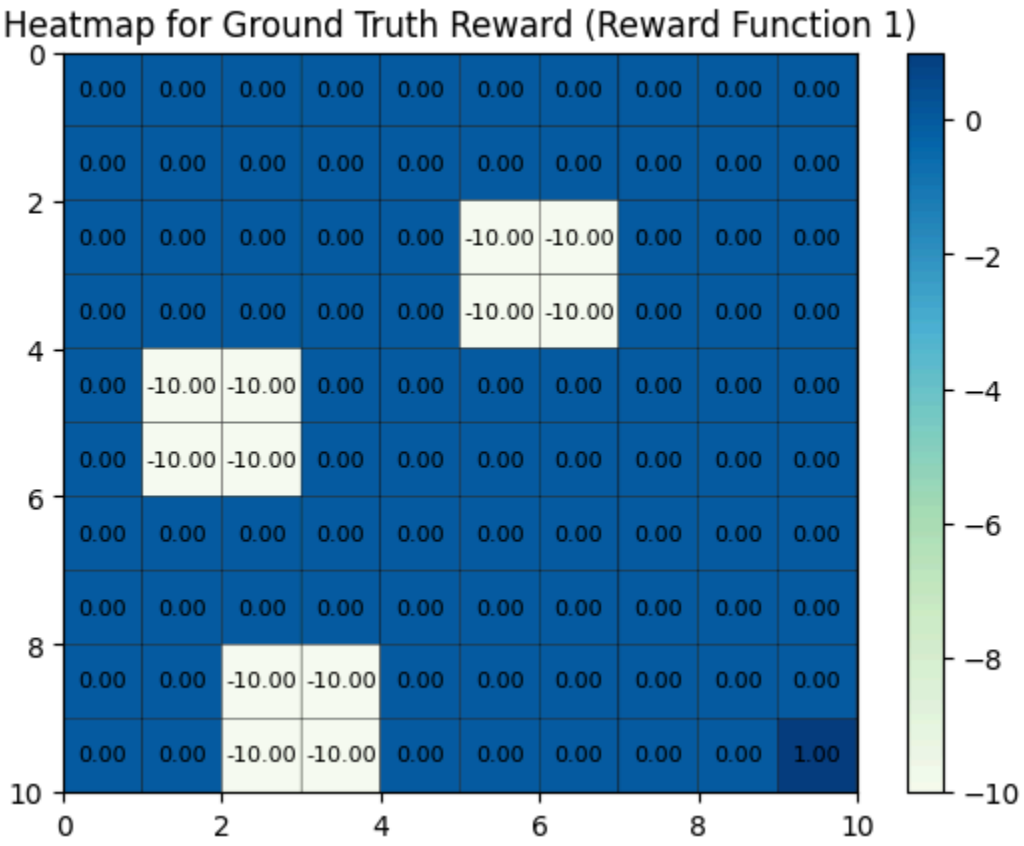
Maximum value of lambda : 0.8216432865731462
Maximum Accuracy : 95.00%



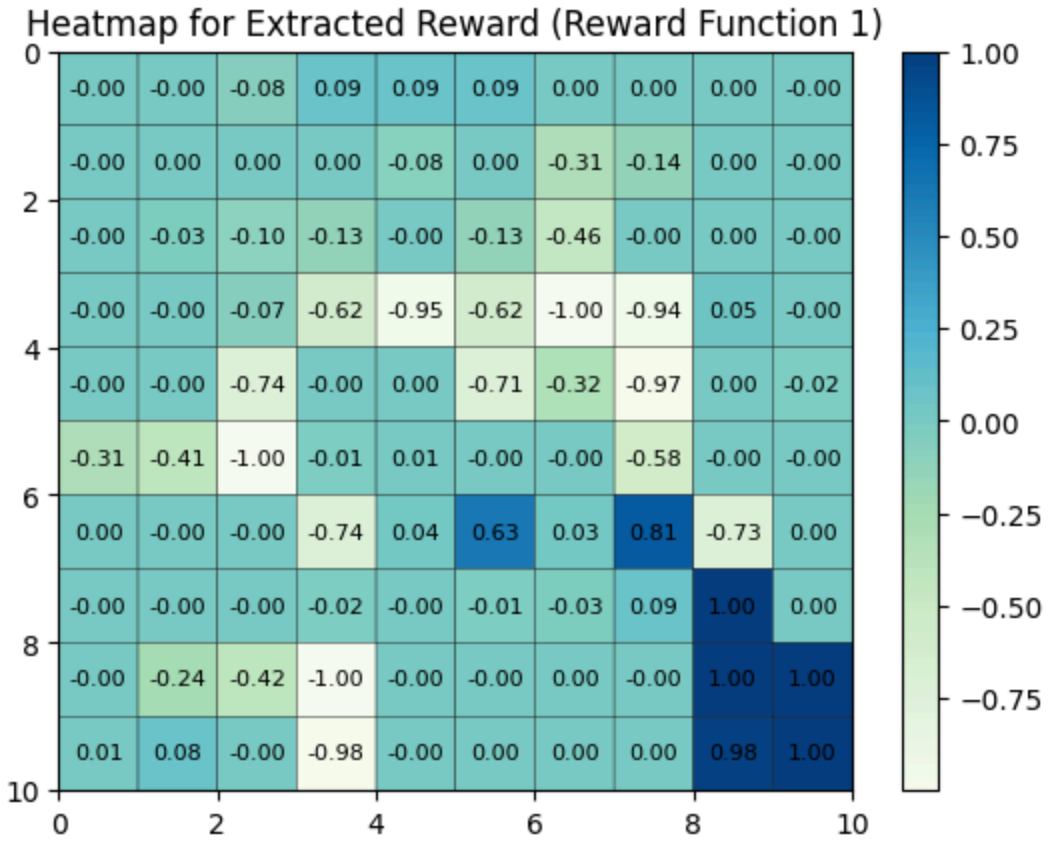
**QUESTION 13:**

**For λ(1) , generate heat maps of the ground truth reward and max the extracted reward. Please note that the ground truth reward is the Reward function 1 and the extracted reward is computed by solving the linear program given by equation 2 with the λ parameter set to λ(1) . In this question, you should have 2 plots. Max**

Ground Truth Reward:




Extracted reward:

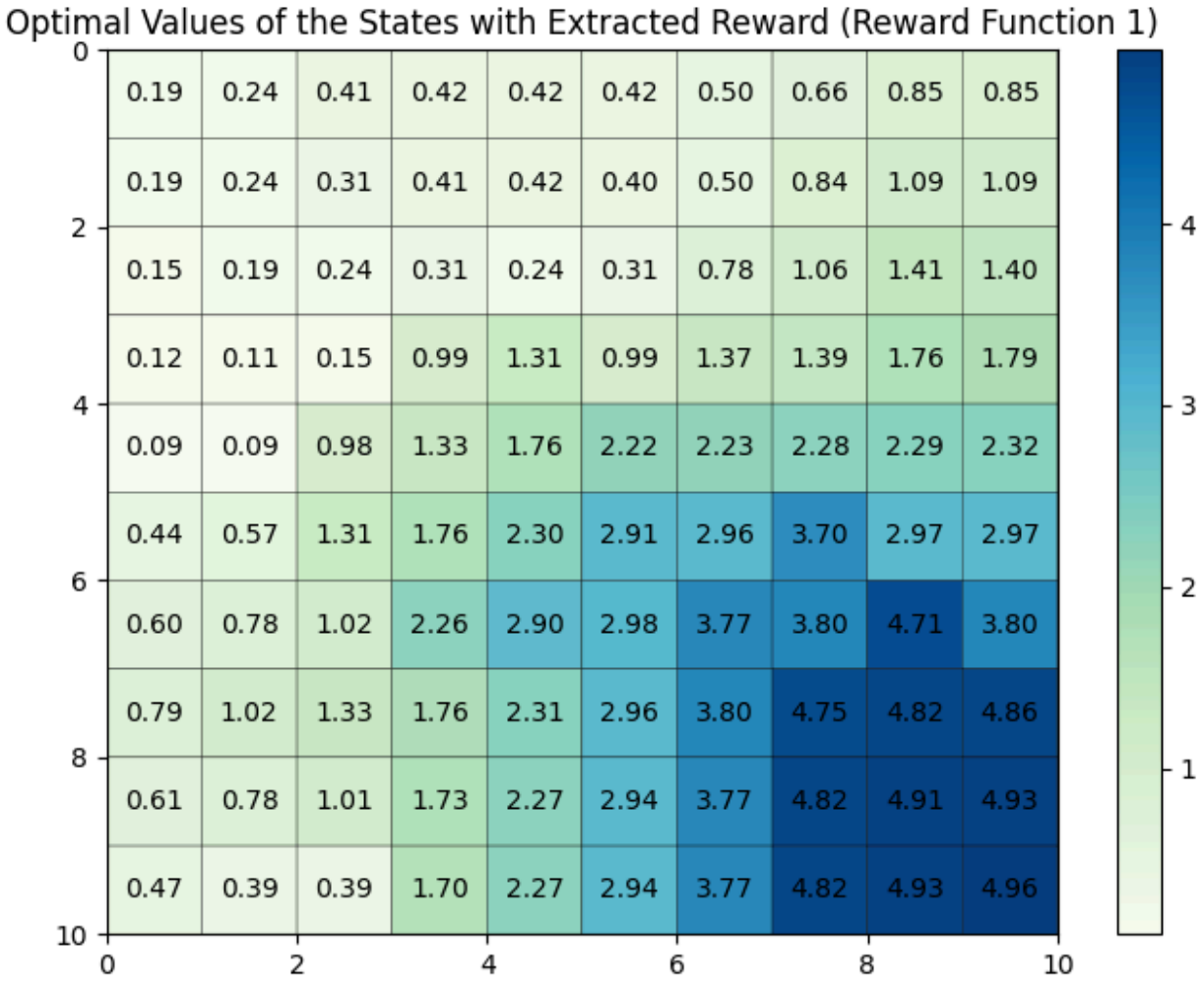



We can observe that the IRL algorithm has correctly identified the regions around state 99 as having high reward, while the regions surrounding the negative-reward blocks have been assigned low immediate rewards. For the extracted reward, as one moves further from state 99, which is the state providing the highest reward, the lower is the immediate reward. In addition, the neighboring states near the blocks yielding negative reward also have a lower reward in the extracted reward heatmap compared to the neighboring states near the state yielding the highest reward. In other words, the extracted rewards are more “continuous” and “spread out” across the state space compared to the actual reward function. This is because the rewards have been extracted from the expert policy and not the ground truth rewards. The expert policy, in turn, is a reflection of the optimal values, which itself is not discrete in nature but changes more gradually compared to the reward function. The use of policy for reward extraction is also the reason for the difference in scales between the extracted reward function and expert reward function.

Learning from policy provides the agent with better information about the state-space in terms of the reward. For example, the closer the agent gets to state 99, the higher will be the rewards it observes. This gradual increase or decrease in reward when approaching “key states” (states with very high or very low reward) helps the agent formulate better policies over rarer rewards, as the agent has greater foresight about the state-space from gradually changing immediate rewards at each state. Such feedback provides the agent with heuristics about which direction to travel to and is specially useful when the state-space is only partially observable and decisions must be made locally based on the immediate rewards.

**QUESTION 14:**

**Use the extracted reward function computed in question 13, to compute the optimal values of the states in the 2-D grid. For computing the optimal values you need to use the optimal state-value function that you wrote in question 2. For visualization purpose, generate a heat map of the optimal state values across the 2-D grid (similar to the figure generated in question 3). In this question, you should have 1 plot.**

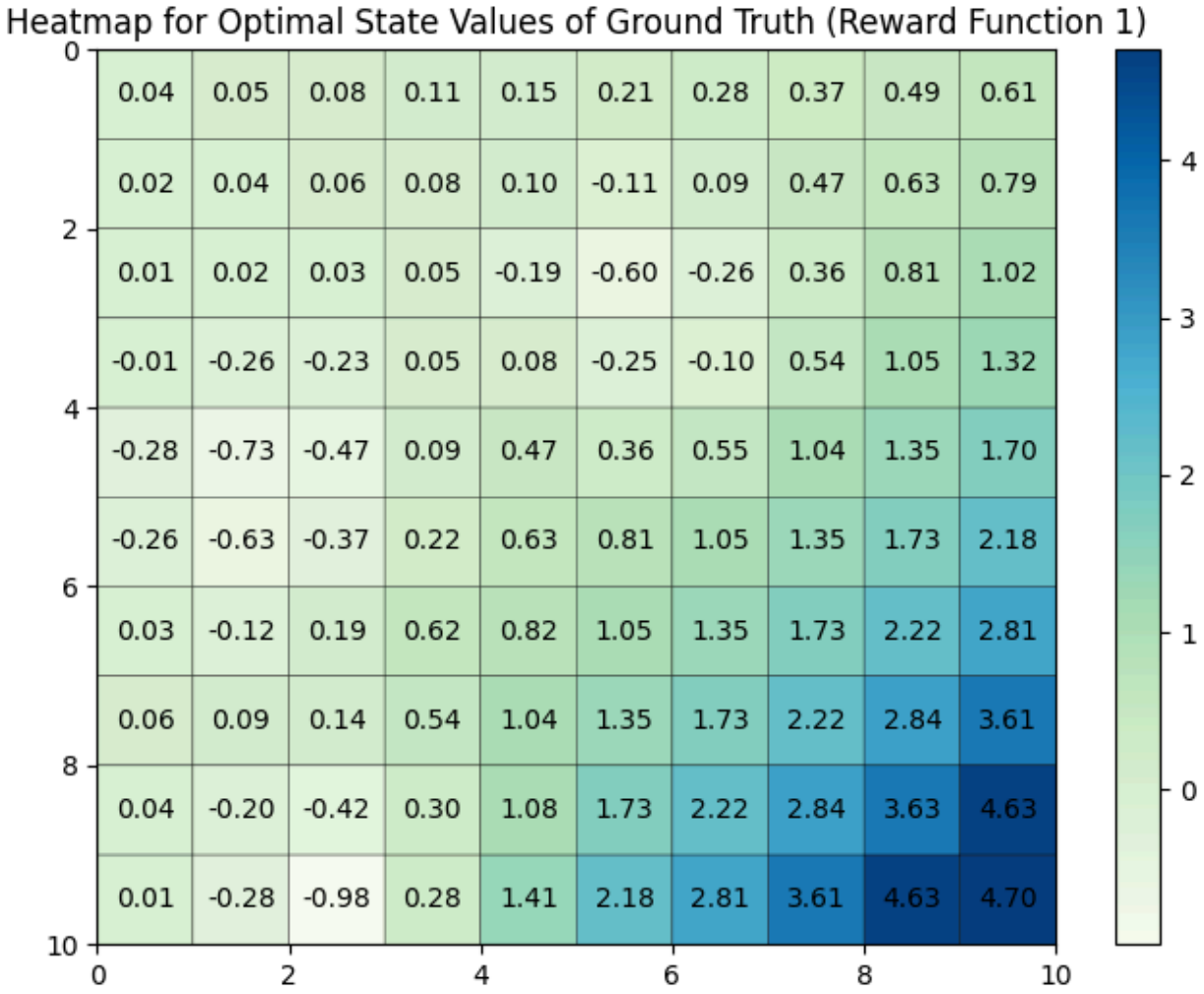



Refer to the heatmap above of the optimal values of the state of the extracted reward function.

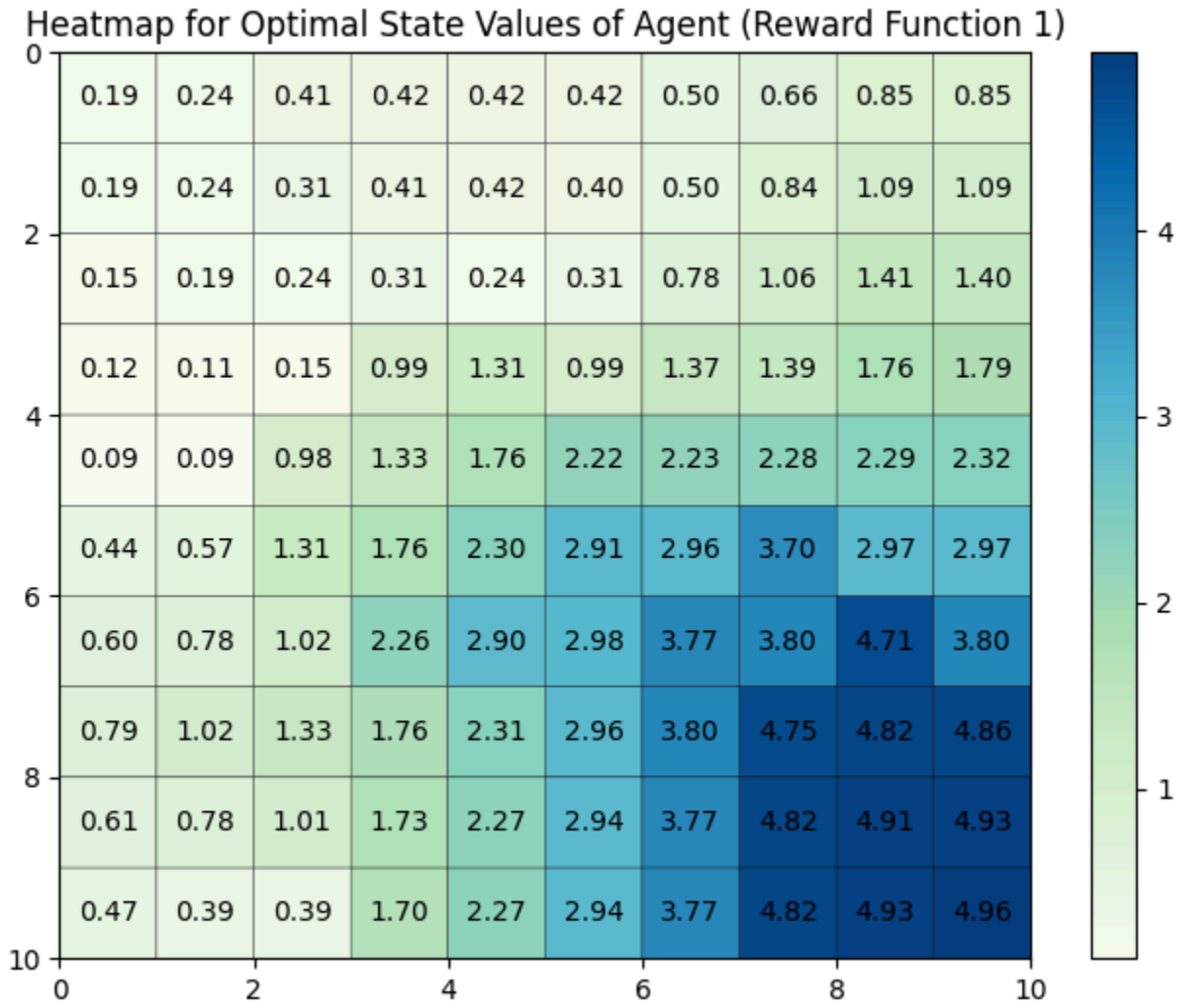
**QUESTION 15:**

**Compare the heat maps of Question 3 and Question 14 and provide a brief explanation on their similarities and differences.**

Heatmap of Question 3:



Heatmap of Question 14:




In both scenarios, we observe a consistent pattern where the vicinity of state 99 exhibits the highest optimal values. This outcome is attributed to state 99 offering the most significant reward within the state space. As we move away from state 99, the optimal values gradually diminish across both plots. Additionally, regions with negative rewards, including neighboring states, exhibit notably low optimal values. The heatmaps derived from Reward Function 1 and those generated in Questions 3 and 14 share similar trends. Notably, the heatmap from Reward Function 1 reveals three distinct blocks of negative rewards, a pattern that is discernible, albeit less pronounced, in the optimal value heatmaps of Questions 3 and 14. These blocks align with expected regions within the state space, indicating a rough correspondence between reward distribution and optimal values. Moreover, regions surrounding states with higher rewards exhibit darker shades in both plots, signifying higher optimal values. Conversely, as an agent navigates towards states with lower rewards, the optimal values of neighboring states progressively decrease. This progressive decay in optimal values emanates from the discount factor present in the Bellman equation, which discounts future rewards. Consequently, this gradual shift is observable in both plots, highlighting a consistent decay in optimal values surrounding states with the highest rewards.

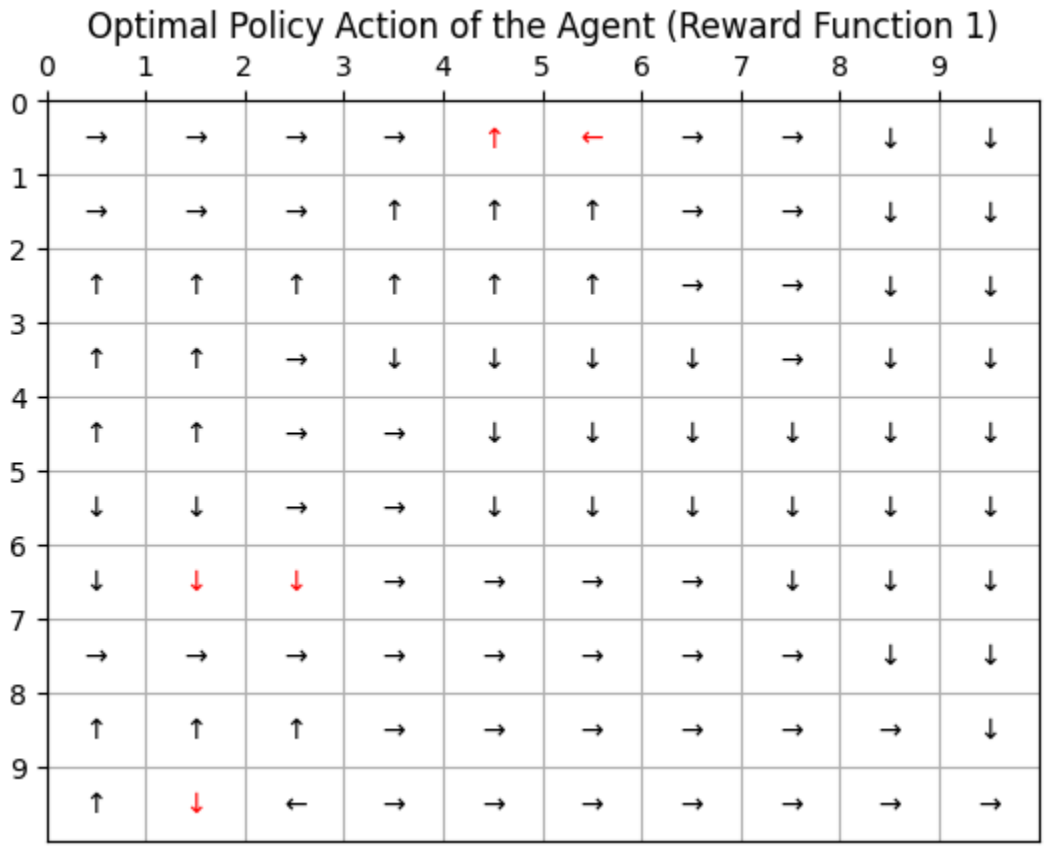
The scaling disparity between the heatmaps of Questions 3 and 14 is evident, with Question 14 displaying a notably smaller range. This discrepancy arises because the heatmap in Question 14 is derived from an

extracted reward, which is computed based on the expert policy rather than the expert reward. Consequently, the scale of optimal state values differs between the two scenarios. In comparison, the regions depicted in the heatmap of Question 3 exhibit clearer delineation and greater uniformity than those in Question 14. This disparity stems from the origin of the heatmaps: Question 3's heatmap originates from the ground truth reward function, featuring predominantly zero rewards aside from three clusters of negative rewards and a single high-reward state. Conversely, Question 14's heatmap is constructed from the extracted reward, which appears more "continuous" and "dispersed" throughout the state space due to its derivation from the expert policy rather than direct reward values. This distinction becomes apparent when examining the reward vector heatmap plots in Question 13, where immediate rewards are assigned to all states. Consequently, most elements in the extracted reward vectors assume nonzero values. As a result, the agent faces less severe penalties in regions with lower rewards in the state space of Question 14 compared to the ground truth scenario of Question 3. Furthermore, in Question 14's state-space, there exists a higher likelihood for the agent to deviate from the optimal state (state 99) due to the rapid accumulation of immediate and localized rewards, potentially overshadowing the maximum achievable reward. In essence, the agent in Question 14 is anticipated to accrue a lower expected reward over the long term compared to the scenario depicted in Question 3. Consequently, the probability of the agent moving off the grid is higher in Question 14's state-space relative to Question 3's.

**QUESTION 16:**

**Use the extracted reward function found in question 13 to compute the optimal policy of the agent. For computing the optimal policy of the agent you need to use the function that you wrote in question 5. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The actions should be displayed using arrows. In this question, you should have 1 plot.**

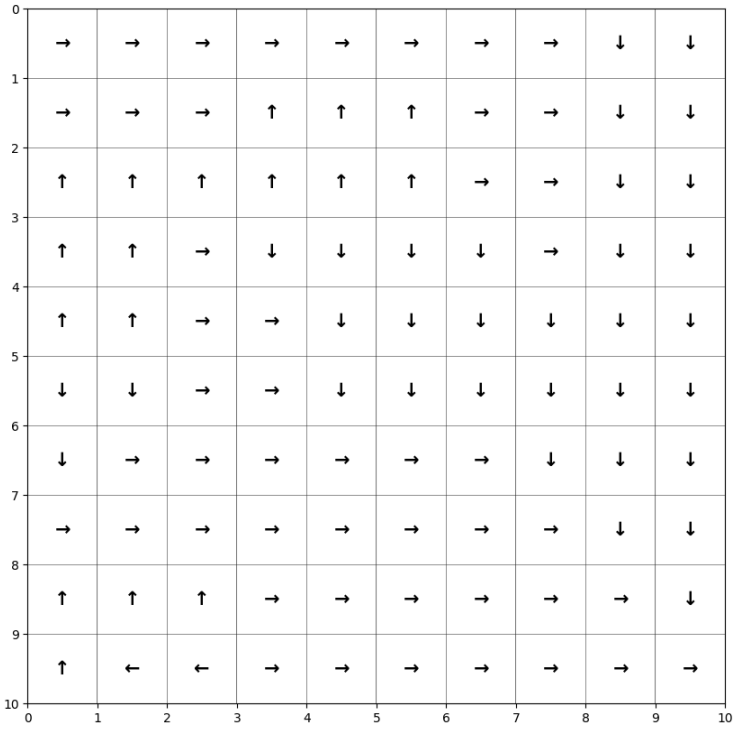
Optimal policy of the agent of extracted reward function:



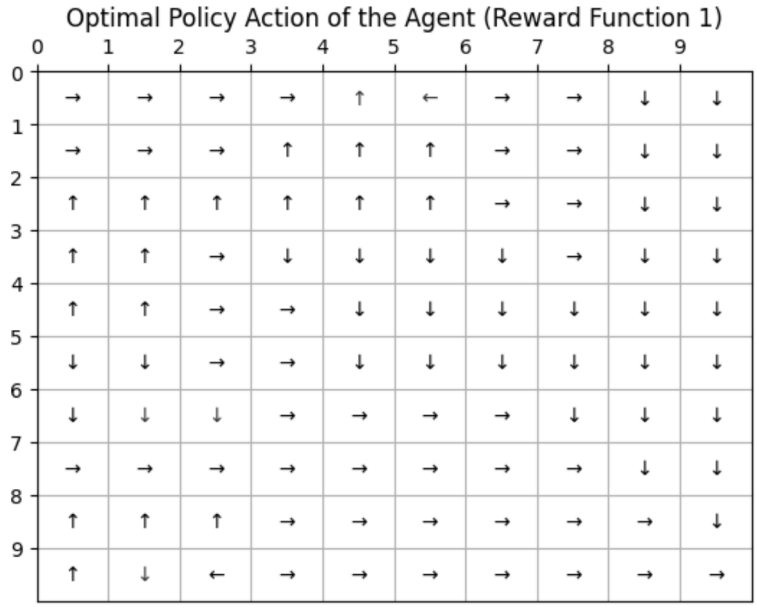
**QUESTION 17:**

**Compare the figures of Question 5 and Question 16 and provide a brief explanation on their similarities and differences.**

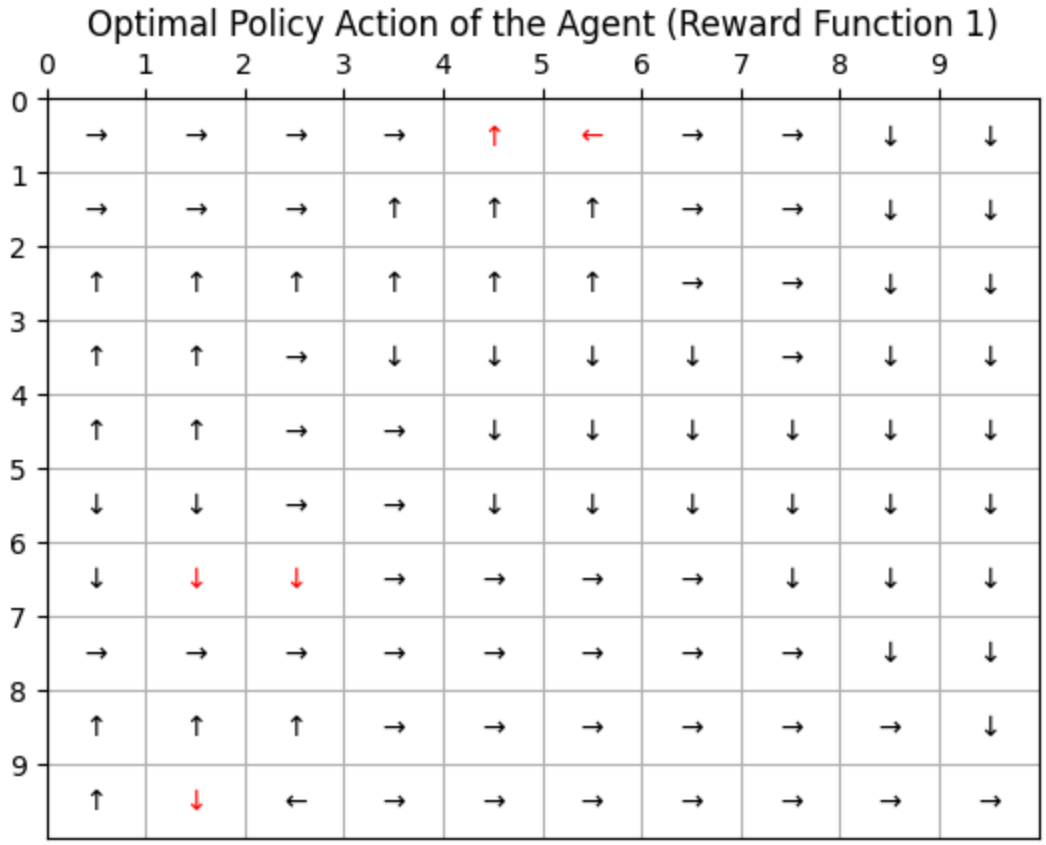
Optimal Policy of Question 5:




Optimal Policy of Question 16:




Difference in Optimal Policy highlighted:




The predominant trend observed in both policy maps is a preference for downward and rightward movements. This inclination can be attributed to the positioning of the most optimal state, state 99, which offers the highest reward and is situated in the lower right corner of the state space. Essentially, in both scenarios, the agent's objective is to navigate towards states offering the highest rewards.

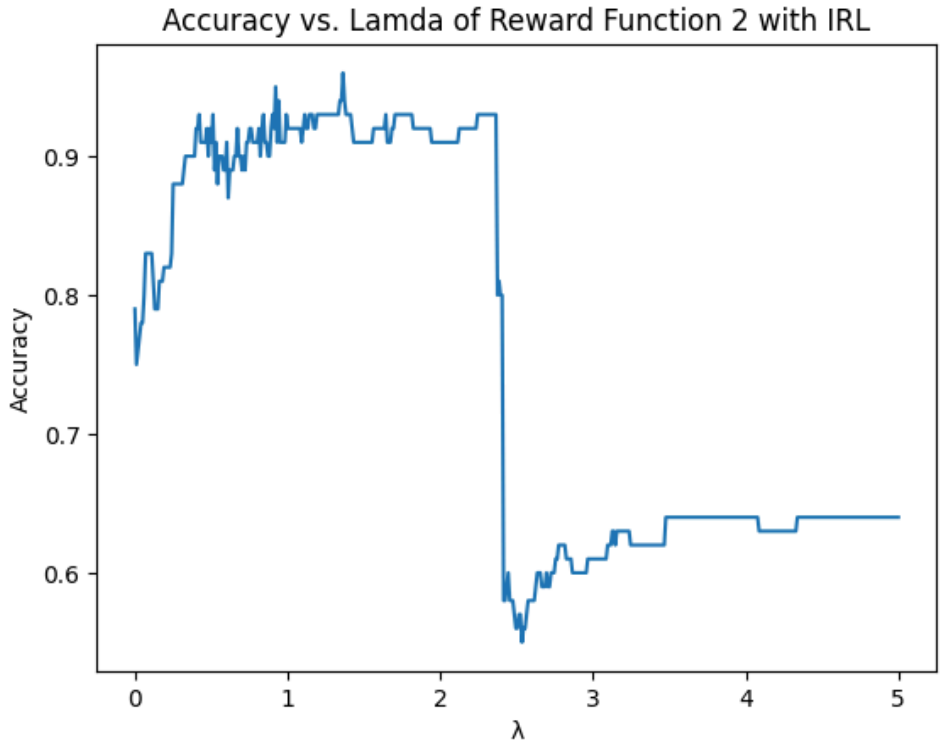
In Question 16, certain actions may result in the agent being blown off the grid, a scenario not encountered by the agent in Question 5. This discrepancy arises due to the state-space characteristics: in Question 14, there is a heightened likelihood of the agent deviating from the optimal state (state 99) compared to Question 3. This increased probability stems from immediate and local rewards potentially diverting the agent away from the optimal trajectory, thereby diminishing the long-term expected reward accumulation. Consequently, in Question 16, the agent is more prone to deviating from the grid or resorting to suboptimal actions over time, contrasting with the more assured progression observed in Question 5.

In Question 5's policy map, the agent is ensured to reach state 99 from any starting point. Conversely, in Question 16, there's a greater risk of the agent encountering grid exit scenarios or becoming ensnared in local optima or deadlock conditions.

Moreover, the actions depicted in Question 16 exhibit more variability and stochasticity compared to those in Question 5. While the actions in Question 5 follow a more orderly and uniform pattern, aimed at directing the agent towards state 99, Question 16's actions display greater diversity, reflecting the increased uncertainty and potential deviations from the optimal path.

**QUESTION 18:**

**Sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ compute OA(s) by following the process described above. For this problem, use the optimal policy of the agent found in question 8 to fill in the OE(s) values. Then use equation 3 to compute the accuracy of the IRL algorithm for this value of λ. You need to repeat the above process for all 500 values of λ to get 500 data points. Plot λ (x-axis) against Accuracy (y-axis). In this question, you should have 1 plot.**




As seen in the above figure, we have plotted the Accuracy (y- axis) v.s. Lamda (x- axis). The graph suggests a pattern in the relationship between the regularization parameter λ and the accuracy of the inverse reinforcement learning (IRL) algorithm. We make the following observations from the graph:

1. **Increasing Accuracy**: Initially, as λ increases, the accuracy of the IRL algorithm also increases. This is because higher values of λ lead to stronger regularization, which helps in preventing overfitting and improves the generalization of the learned model.
2. **Sharp Dip**: After a certain point, there's a sharp dip in accuracy as λ continues to increase. This dip indicates that too much regularization starts to have a detrimental effect on the accuracy. It suggests that the model is being overly constrained by the regularization term, leading to a loss in performance.
3. **Stable Values**: Following the sharp dip, the accuracy stabilizes and remains relatively constant as λ further increases. This stability suggests that beyond a certain threshold, additional regularization does not significantly impact the accuracy of the IRL algorithm. The model has reached a point where it is sufficiently regularized to avoid overfitting without sacrificing too much accuracy.

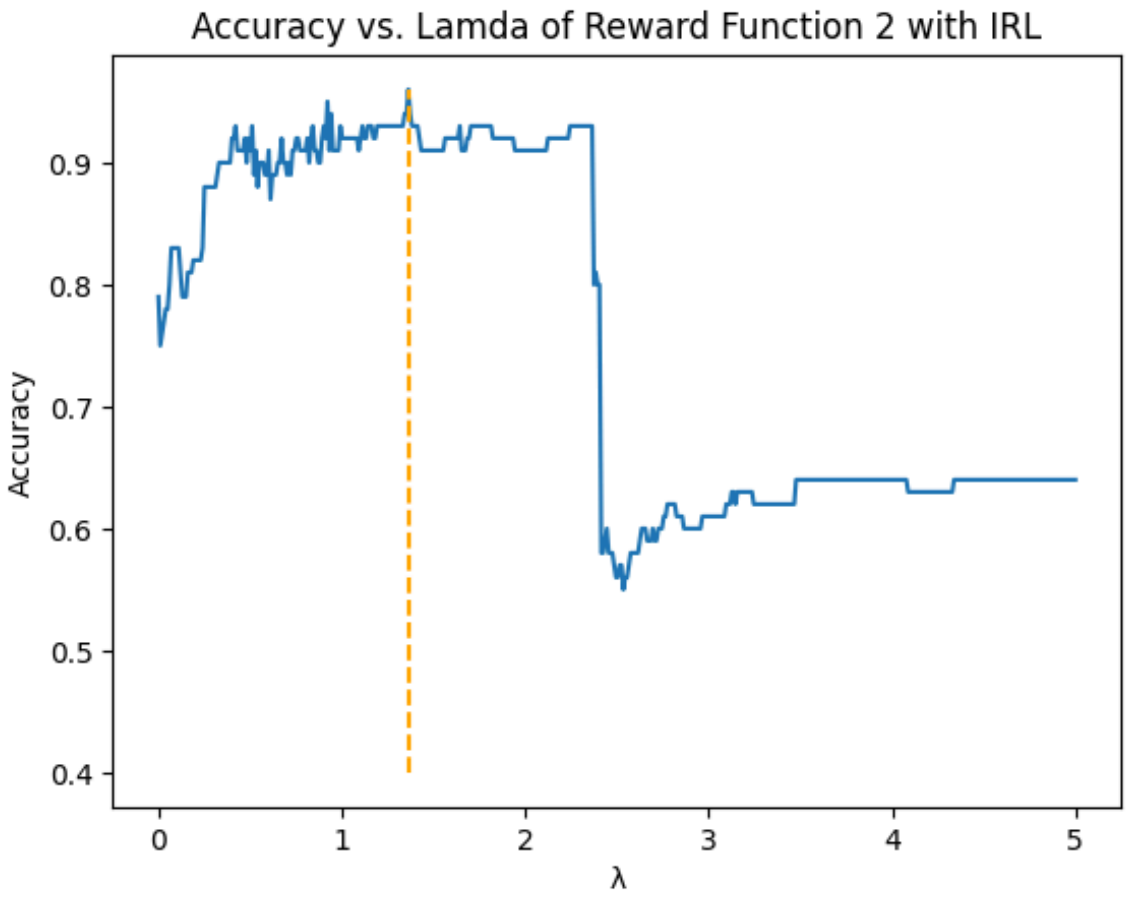
The above pattern is expected because Lamda acts as a regularizer (L1 or Lasso normalization), encouraging simpler reward vectors. Finite values of improves robustness, succinctness and transferability of the extracted reward function such that it generalizes well when extracting the optimal policy, which explains the initial rise in accuracy. Large values of will yield reward vectors that underfit the tasks and are not expressive enough to derive the optimal global policy, with increased risk of being stuck in a locally optimum policy.

The above pattern is common when tuning regularization parameters in machine learning models. There's often a trade-off between model complexity (captured by the data fidelity term) and regularization (captured by the regularization term). The goal is to find the right balance that maximizes accuracy while preventing overfitting.

**QUESTION 19:**

**Use the plot in question 18 to compute the value of λ for which accuracy is maximum. For future reference we will denote this value as λ(2) . Please report λ(2) max**

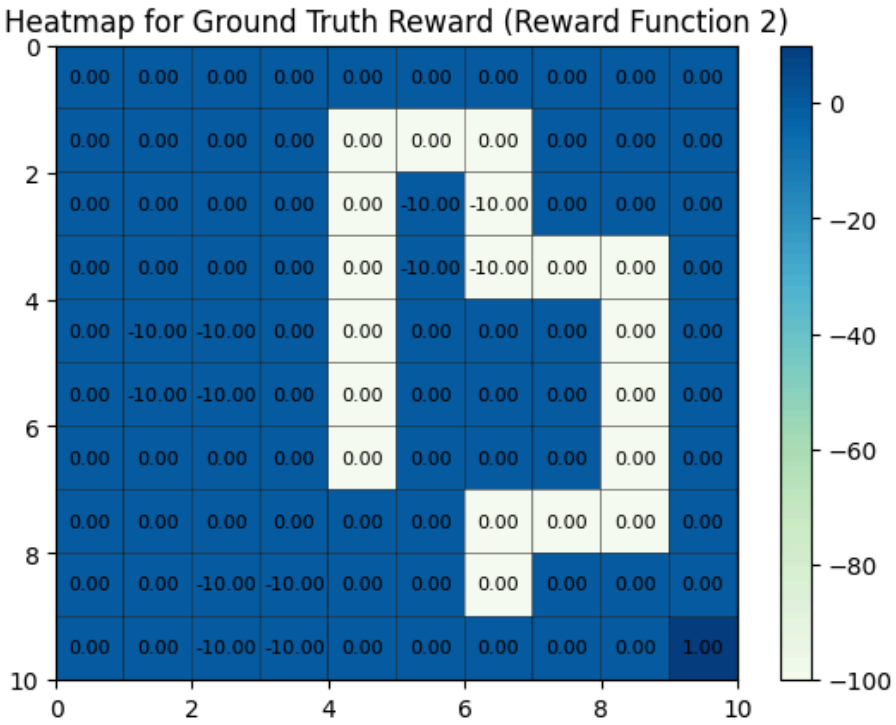
**Maximum value of lambda $\lambda(2)_{max}$ : 1.3627254509018036**
**Maximum Accuracy : 0.96**



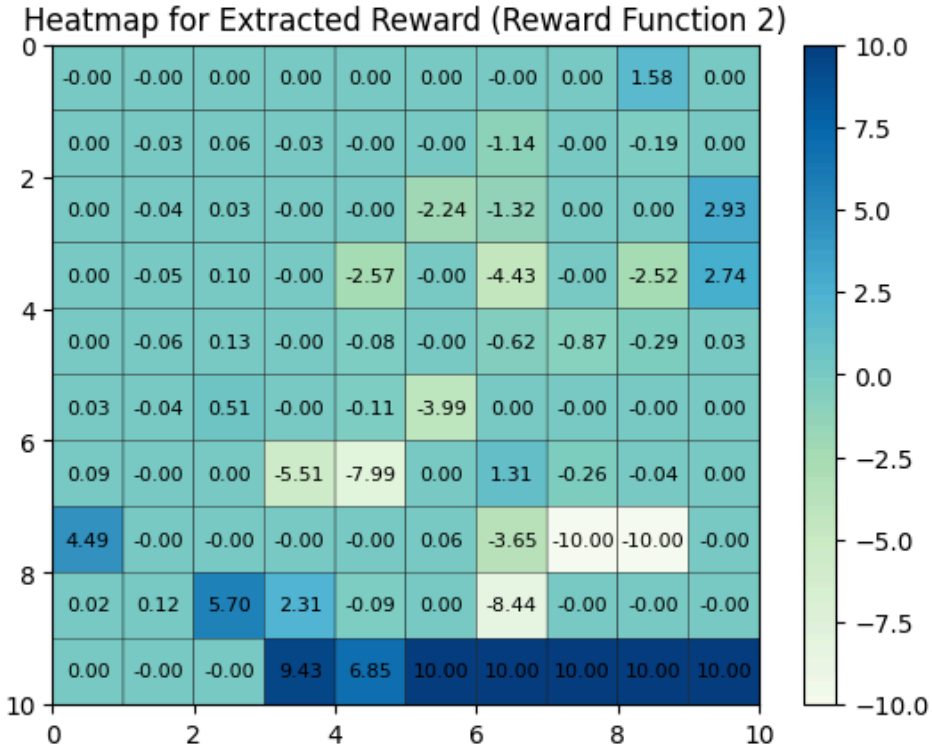
**QUESTION 20:**

**For λ(2) , generate heat maps of the ground truth reward and max the extracted reward. Please note that the ground truth reward is the Reward function 2 and the extracted reward is computed by solving the linear program given by equation 2 with the λ parameter set to λ(2) . In this question, you should have 2 plots.**

**Ground Truth Reward:**




**Extracted Reward:**

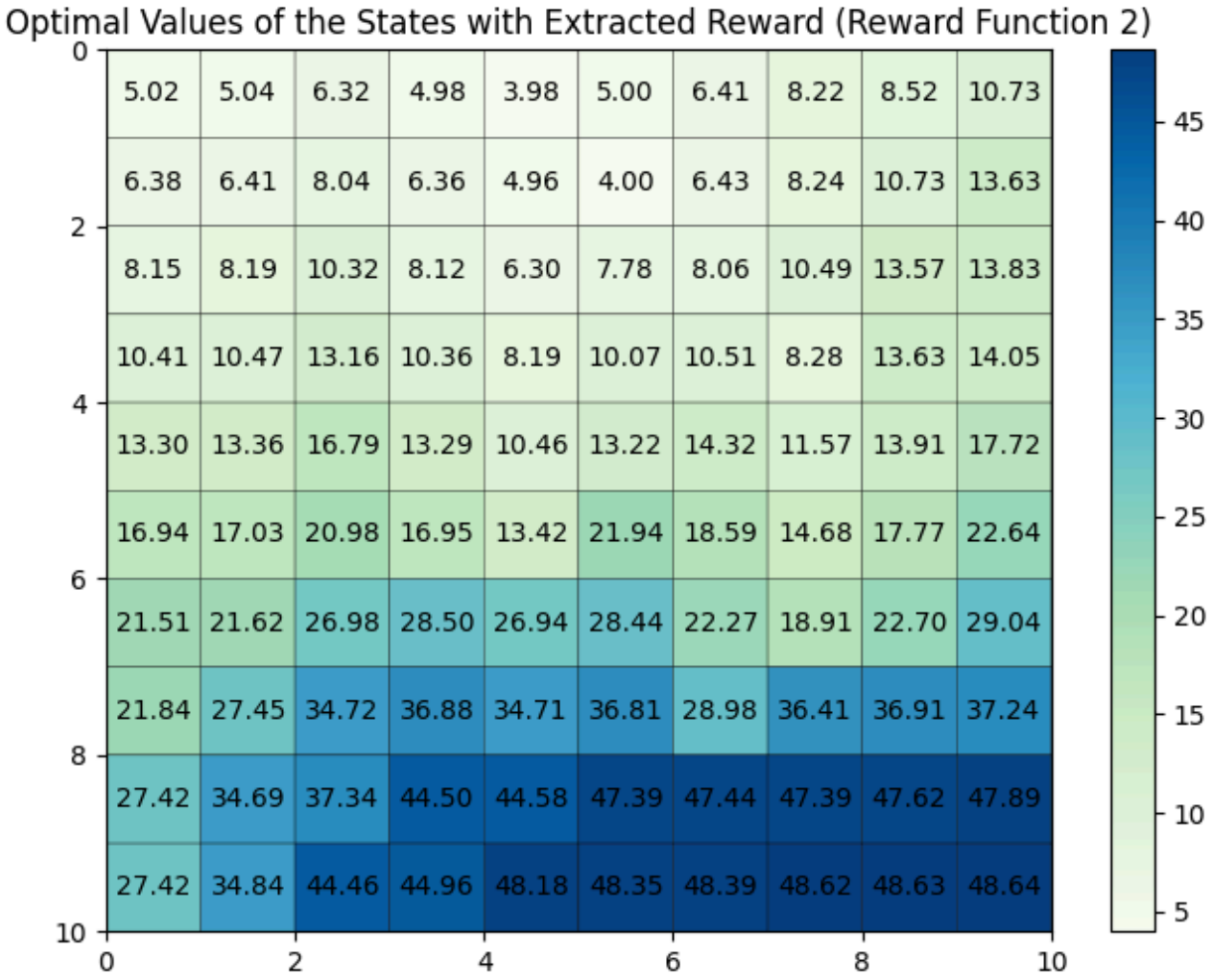


The above figures illustrate that the rewards extracted by the IRL algorithm exhibit a smoother distribution across the state space compared to the actual reward function. This difference arises because the extracted rewards are derived from the expert policy rather than directly from the ground truth rewards. The expert policy, influenced by optimal values, undergoes gradual changes unlike the discrete nature of the reward function. Consequently, the scales of the extracted and expert reward functions differ.

Furthermore, the IRL algorithm accurately identifies regions surrounding state 99 as having high rewards and the negative-reward chain as having low rewards. In the extracted reward heatmap, as one moves away from state 99, the immediate reward diminishes. Similarly, neighboring states near the negative-reward chain exhibit lower rewards in the extracted reward heatmap compared to those near the highest-reward state. However, a notable difference is the identification of another high-reward region near states 28 and 38, surpassing state 99. This discrepancy arises because the reward function alone doesn't fully indicate long-term returns; thus, learning from the expert policy, which incorporates optimal state values, enhances the algorithm's understanding of the state-space. This learning process offers insights into navigating the state-space effectively, especially when decisions are based on immediate rewards in partially observable environments.

**QUESTION 21:**

**Use the extracted reward function computed in question 20, to compute the optimal values of the states in the 2-D grid. For computing the optimal values you need to use the optimal state-value function that you wrote in question 2. For visualization purpose, generate a heat map of the optimal state values across the 2-D grid (similar to the figure generated in question 7). In this question, you should have 1 plot.**

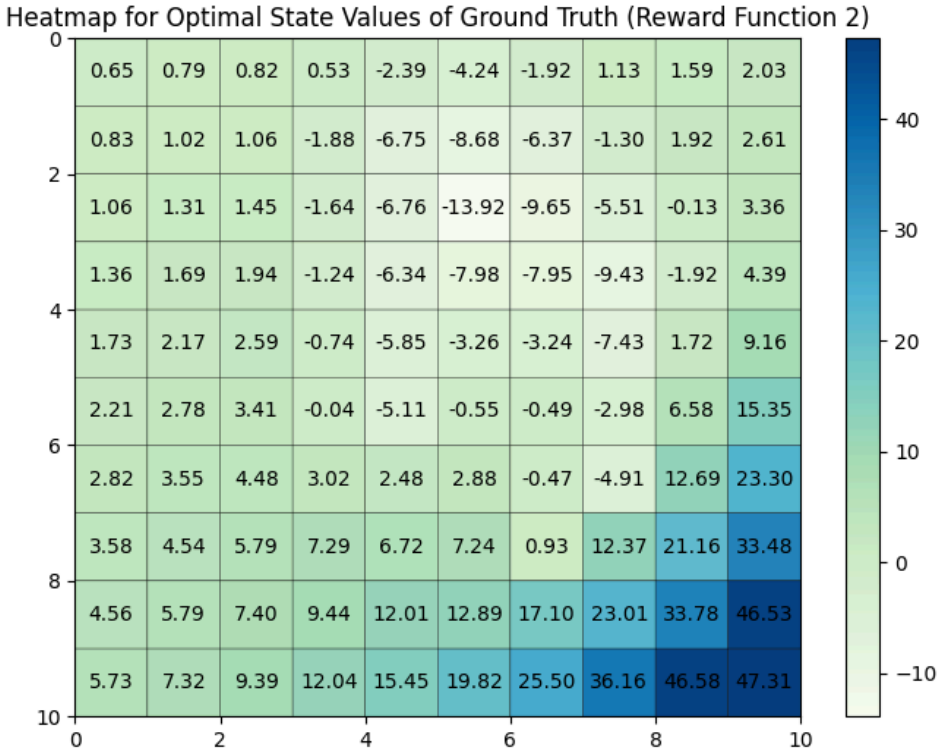



The above heatmap shows the optimal state-values from the extracted reward function with using value iteration.

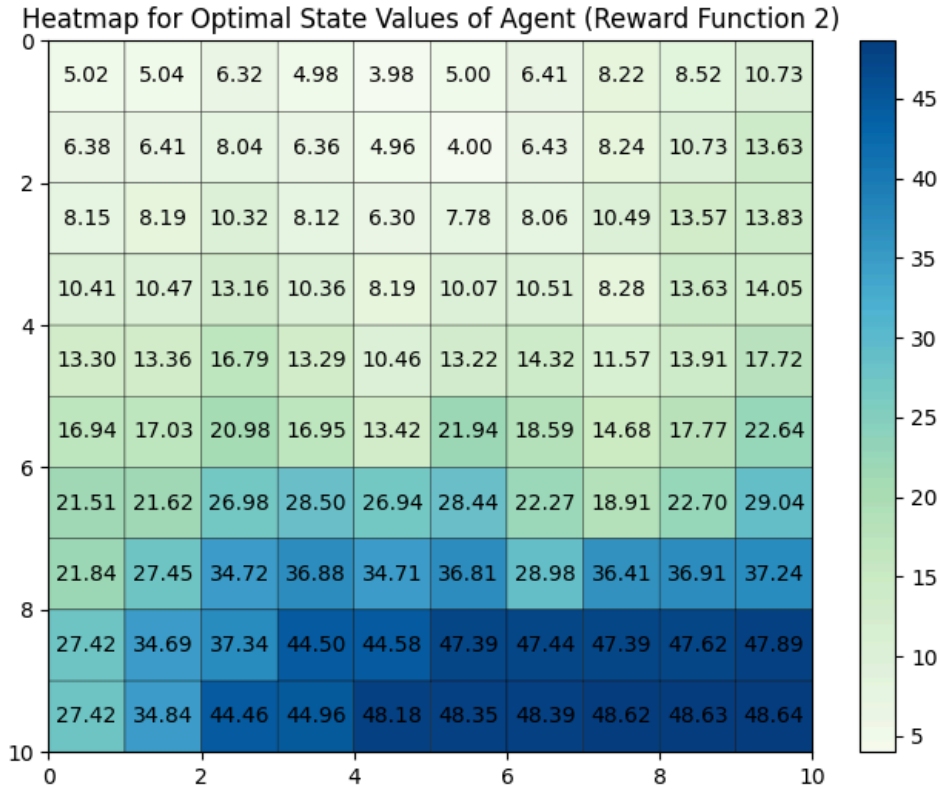
**QUESTION 22:**

**Compare the heat maps of Question 7 and Question 21 and provide a brief explanation on their similarities and differences.**

**Heatmap from Question 7**




**Heatmap from Question 21**



In this comparison, we're examining the heatmaps from Question 7 and Question 21, focusing on both similarities and differences:

**Similarities:**

1. The area with negative rewards (representing the lowest optimal values) is roughly consistent and located similarly in both heatmaps.
2. Proximity to state 99 corresponds to high optimal values in both plots, although it's not the highest in one of them. State 99 yields the highest reward, causing values to gradually decrease as one moves away from it in both plots.
3. When we observe the heatmaps from both plots, we can notice a gradual change in optimal values as the agent navigates through different states. As the agent moves towards states with low rewards, the neighboring states exhibit a progressive decrease in optimal values. Conversely, when the agent heads towards states with higher rewards, the optimal values of surrounding states gradually increase. This trend occurs because of the discount factor incorporated into the Bellman equation. This factor essentially downgrades the significance of future rewards, influencing the agent's decision-making process. Consequently, the observed gradual decay or increase in optimal values around states with low or high rewards respectively illustrates how the agent strategically adapts its actions to maximize long-term cumulative rewards while considering the discounted nature of future rewards. While this phenomenon is evident in both plots, it appears more prominently in Question 21 due to the specific characteristics of the extracted reward function.

**Differences:**

1. In Question 21's heatmap, we notice a shift in the distribution of optimal values compared to the region near state 99. Although state 99 still exhibits high optimal values, it's not the highest in this case. Instead, the highest optimal values are now found in the lower-left corner of the state-space. This change is attributed to the limitations of relying solely on the reward function as an indicator of long-term expected return. While the reward function provides valuable information, it's essential to complement it with insights from the transition matrix and optimal state values. In the context of Inverse Reinforcement Learning (IRL), which learns the reward function from expert policies, this integration of various factors allows for a more nuanced understanding of the state-space. By learning from expert policies, which encapsulate optimal expert state values, the IRL algorithm gains valuable heuristics about the state-space beyond what traditional value iteration methods offer. One significant advantage of learning from policy is that it provides the agent with richer information about the state-space concerning rewards. For instance, as the agent approaches state 99, it observes higher rewards, contributing to a gradual increase in reward. This gradual change in reward as the agent navigates towards "key states" with very high or low rewards enables the agent to formulate more effective policies, especially in scenarios where the state-space is only partially observable, and decisions must be made based on immediate rewards. In essence, this feedback loop, where the agent gains insights from gradually changing immediate rewards at each state, equips it with valuable heuristics about which direction to move towards, enhancing its decision-making process, particularly in complex and partially observable environments.

2. The difference in scaling between the heatmaps from Question 7 and Question 21 can be attributed to the methodology used to generate them. In Question 7, the heatmap is derived directly from the ground truth reward function. This means that the values in the heatmap represent the actual rewards associated with each state in the environment. Since the ground truth reward function provides a direct measure of rewards, the range of optimal state values in the heatmap corresponds directly to the range of rewards

specified in the reward function. In contrast, in Question 21, the heatmap is generated from the extracted reward, which is calculated based on the expert policy rather than the expert reward. This means that the values in the heatmap reflect the optimal values obtained from the expert policy, which may not necessarily align perfectly with the rewards specified in the ground truth reward function. Consequently, the range or scale of optimal state values in the heatmap from Question 21 may differ from that of Question 7. The scaling difference arises because the extracted reward is influenced by the expert policy's decisions, which could lead to variations in the range of optimal state values compared to the ground truth reward function. Therefore, while both heatmaps represent optimal values, the source of the data (ground truth reward function versus extracted reward from expert policy) can result in different scaling characteristics between the two heatmaps.

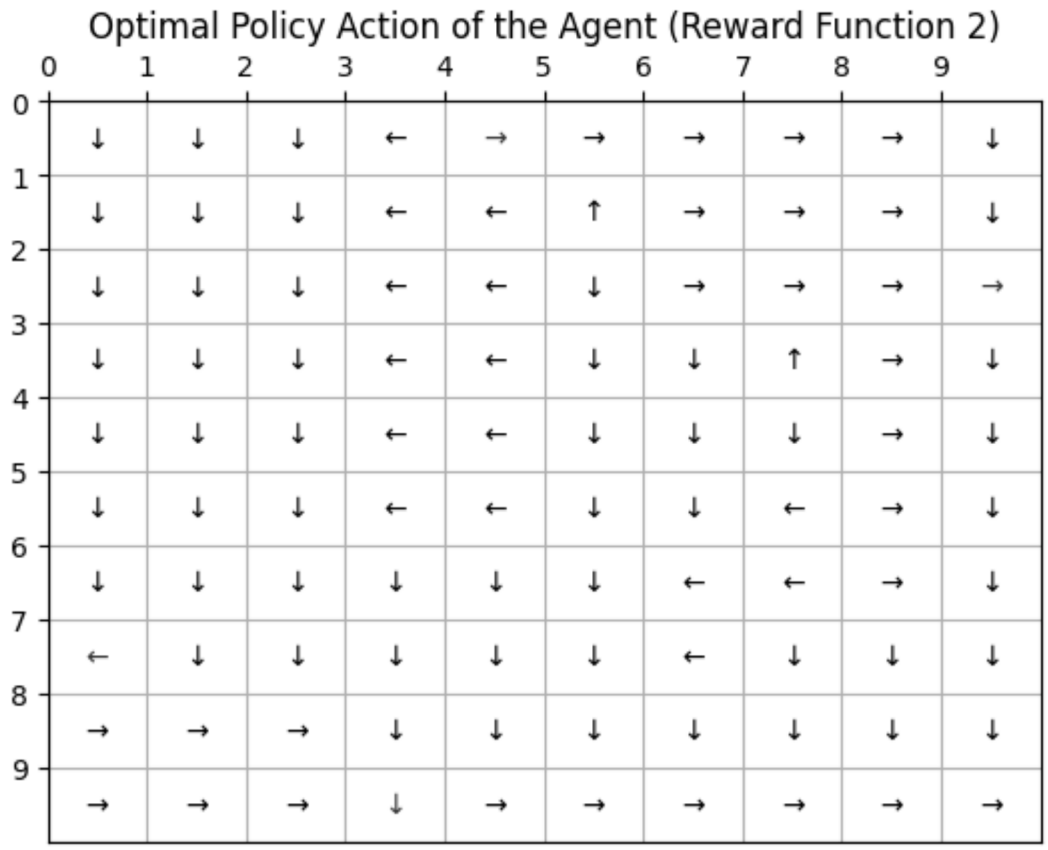
3. In Question 7, the heatmap is derived directly from the ground truth reward function, which typically assigns 0 reward to most states, except for specific regions like the negative-reward snake-like chain and perhaps a single high-reward state. As a result, the regions in the heatmap from Question 7 are well-defined and homogeneous, with clear boundaries between areas of different reward values. This is because the ground truth reward function provides a clear and discrete assignment of rewards to states. On the other hand, in Question 21, the heatmap is generated from the extracted reward, which is influenced by the decisions of the expert policy rather than the direct rewards specified in the ground truth reward function. As seen in the reward vector heatmap plots from Question 20, where immediate rewards are assigned to all states, the extracted reward tends to be more continuous and spread out across the state space. This means that most states have nonzero rewards, contributing to a more continuous distribution of optimal values in the heatmap from Question 21. Therefore, the differences in homogeneity and definition between the two heatmaps stem from the source of the reward information used to generate them. The ground truth reward function results in well-defined regions with discrete reward values, while the extracted reward from the expert policy leads to a more continuous and spread-out distribution of rewards, influencing the homogeneity and definition of regions within the heatmap.

4. In Question 7, where the heatmap is derived from the ground truth reward function, the agent faces a more stringent penalty near regions with lower rewards. This means that the agent is discouraged from moving towards states with lower rewards, as the lower immediate rewards and harsh penalties associated with those regions push the agent away from them. Consequently, the agent is more inclined to seek out optimal regions where the long-term expected reward is maximized. However, in Question 21, where the heatmap is generated from the extracted reward based on the expert policy, the penalties near regions with lower rewards are less severe compared to Question 7. This means that the agent is not as strongly discouraged from exploring states with lower immediate rewards, as the penalties are milder. As a result, there's a higher probability for the agent to remain in or move towards these suboptimal regions, where immediate and local rewards might seem attractive but ultimately lead to a lower long-term expected reward. In essence, the differences in how the agent is penalized near regions with lower rewards between Question 21 and Question 7 influence the agent's decision-making process. In Question 21, the agent is more likely to be lured away from optimal regions by the allure of immediate rewards, leading to a lower expected reward in the long run compared to the agent in Question 7. This difference underscores the importance of understanding how different reward structures can impact an agent's behavior and ultimately its performance in a given environment.

**QUESTION 23:**

**Use the extracted reward function found in question 20 to compute the optimal policy of the agent. For computing the optimal policy of the agent you need to use the function that you wrote in question 9. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The actions should be displayed using arrows. In this question, you should have 1 plot.**

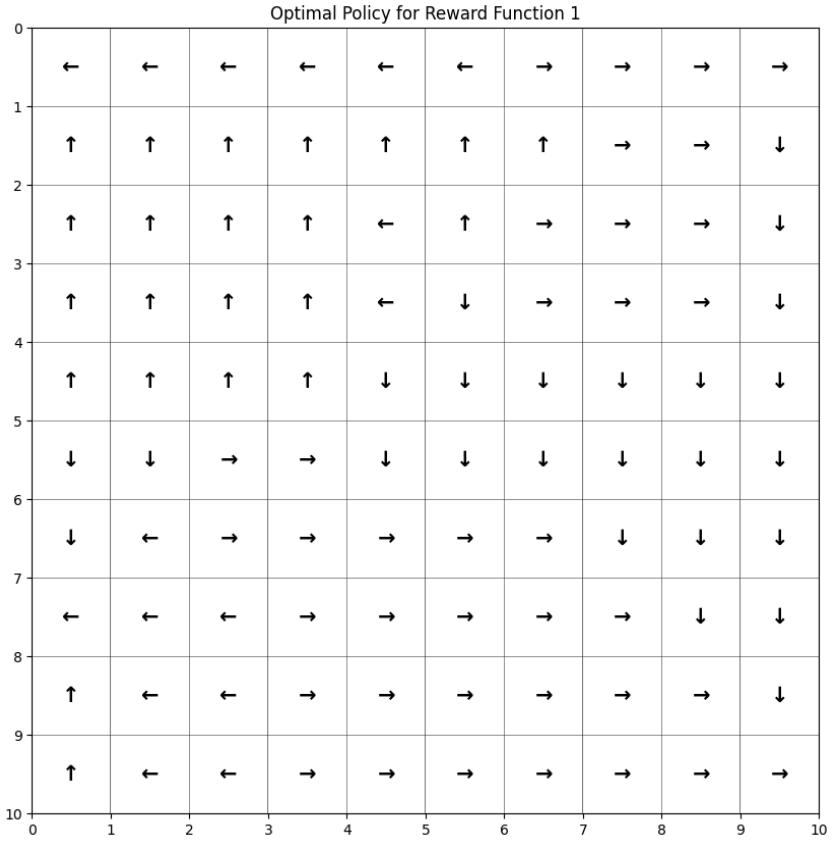
The optimal policy for reward function 2 is given below:



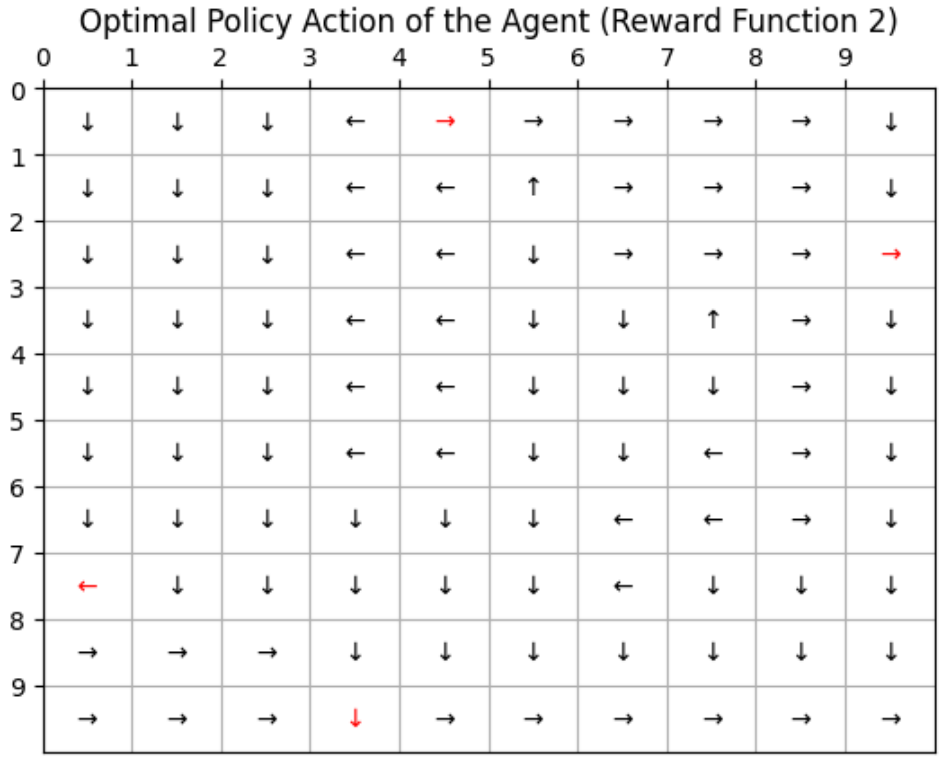
**QUESTION 24:**

**Compare the figures of Question 9 and Question 23 and provide a brief explanation on their similarities and differences.**

**Optimal Policy Question 9:**




**Optimal Policy Question 23:**



Comparison between the action plots in Question 9 and Question 23, identifying both similarities and differences:

**Similarities:**

1. In both scenarios, the agent demonstrates a tendency to navigate away from areas characterized by low rewards or suboptimal values, while aiming to approach regions associated with high rewards or optimal values.

2. Divergence of some arrows, indicating movement away from each other, is observed in both cases. This phenomenon may arise from encountering incompatible states or conditions within the environment.

3. Actions are distributed evenly across all four cardinal directions (up, down, left, and right) in both plots. This implies that the agent employs a strategy of exploring all possible directions to reach optimal states efficiently.

**Differences:**

1. In Question 23, compared to Question 9, we observe the presence of two distinct high-reward regions, as indicated by the convergence of arrows towards these areas. This contrast arises because the Inverse Reinforcement Learning (IRL) algorithm, utilized in Question 23, identifies an additional high-reward region located in the lower left corner around states 28 and 38. This finding is consistent with the insights obtained from both Question 20 and Question 22. The identification of this additional high-reward region underscores the limitations of relying solely on the reward function as an indicator of long-term expected return. Instead, it emphasizes the importance of integrating information from the transition matrix and optimal state values alongside the reward function. In the context of IRL, the algorithm learns the reward function from expert policies, which in turn reflect optimal expert state values. This learning process equips the IRL algorithm with enhanced insights and heuristics about the state-space compared to conventional value iteration methods. In summary, the presence of two high-reward regions in Question 23, as opposed to one in Question 9, highlights the IRL algorithm's ability to discern nuanced patterns within the state-space, ultimately leading to a more comprehensive understanding of optimal regions and improved decision-making capabilities.

2. In Question 23, there are instances where certain actions may lead the agent to move off the grid, a scenario not observed for the agent in Question 9. This difference arises due to the nature of the state-space in Question 21, which influences the agent's decision-making process differently compared to Question 7. In Question 21, where the state-space is influenced by the extracted reward based on the expert policy, the agent faces a higher probability of not moving towards optimal regions. This is because immediate and local rewards can sometimes lure the agent towards suboptimal directions, overshadowing the consideration of the maximum achievable long-term reward. Consequently, the agent may accumulate a lower expected reward over time in Question 21 compared to Question 7. This discrepancy in expected reward between the two scenarios can lead to a higher likelihood of the agent moving off the grid or taking suboptimal actions in the long run, particularly evident in Question 23. The agent's decision-making process in Question 23 may be influenced by the allure of immediate rewards, causing it to deviate from optimal paths or even venture beyond the boundaries of the grid. Overall, the difference in the state-space characteristics between Question 21 and Question 7 impacts the agent's behavior and decision-making strategy, ultimately influencing its long-term performance and propensity to stray from optimal trajectories, particularly noticeable in the actions taken in Question 23.

3. In Question 23, we observe that some actions point towards each other, indicating local optima in the policy plots. This phenomenon results in what's known as a deadlock condition, where the agent continually oscillates between two states without making progress. This occurrence arises due to the gradual changes in immediate rewards among neighboring states. When the immediate rewards between two neighboring states are very similar and non-zero, the agent becomes trapped in a cycle of moving back and forth between these states, especially if the exploration probability is low. This deadlock condition is absent in the policy map of Question 9 because most rewards in the state space are zero. This absence of nonzero rewards enhances the agent's foresight, allowing it to propagate the influence of high-reward states to neighboring states much more effectively than in Question 23. In essence, the values in the state-space of Question 9 exhibit less noise compared to Question 23. This reduced noise enables the agent in Question 9 to discern the correct direction from a further distance, avoiding the trap of oscillating between states. Consequently, the absence of deadlock conditions in Question 9 enhances the efficiency of the agent's decision-making process and its ability to navigate towards optimal states more effectively.

4. In Question 9, where most rewards are zero, the agent can reliably reach the optimal state from any starting point due to clear decision-making signals. However, in Question 23, the presence of non-zero rewards introduces uncertainty. This increases the risk of the agent being led off the grid or getting stuck in suboptimal paths, making it less certain to reach the optimal state from any starting point.

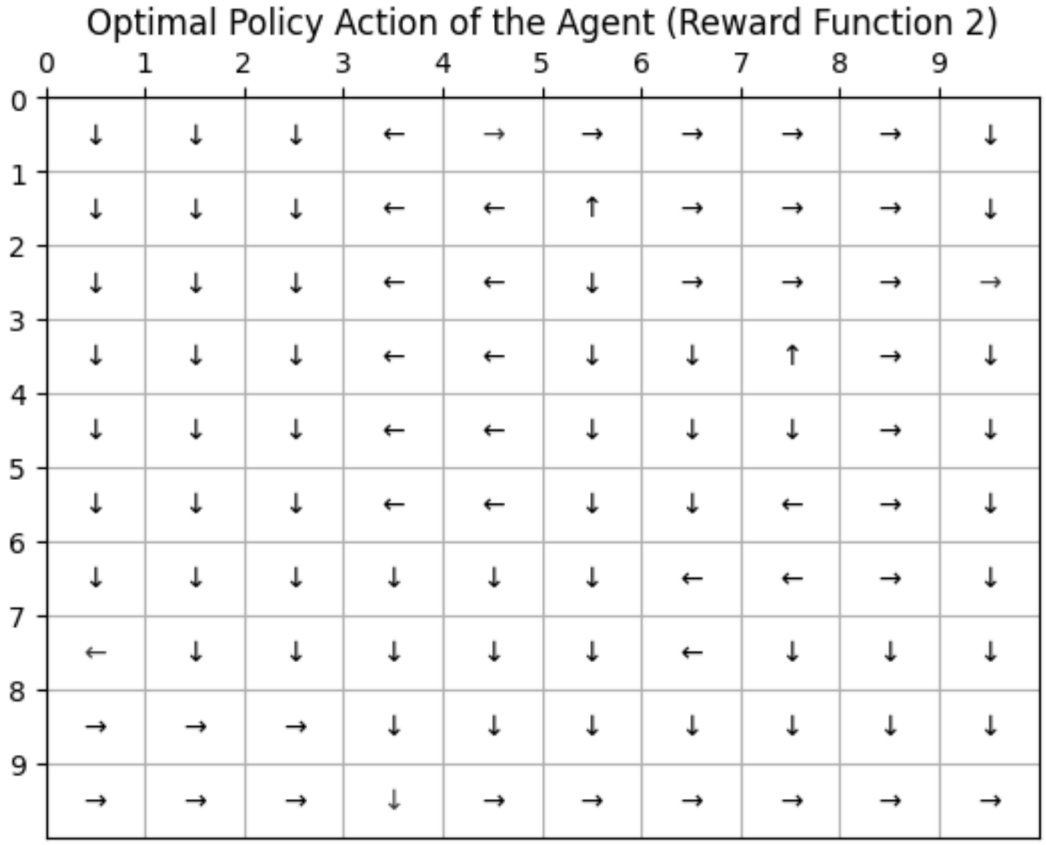
5. In Question 9, where most rewards are zero, the agent's actions are ordered and directed towards the optimal state in a predictable manner. However, in Question 23, the presence of non-zero rewards introduces variability and randomness into the agent's actions, leading to a less deterministic path towards the optimal state.

**QUESTION 25:**

**From the figure in question 23, you should observe that the optimal policy of the agent has two major discrepancies. Please identify and provide the causes for these two discrepancies. One of the discrepancy can be fixed easily by a slight modification to the value iteration algorithm. Perform this modification and re-run the modified value iteration algorithm to compute the optimal policy of the agent. Also, recompute the maximum accuracy after this modification. Is there a change in maximum accuracy? The second discrepancy is harder to fix and is a limitation of the simple IRL algorithm.**

The first discrepancy one can observe is that the policy of some states at the edges or corner is moving off the grid instead of following the flow pointed toward the positive reward direction. This is because the way we designed the transition probability as well as the value iteration function might potentially mislead those states to prefer to take the policy that will let them stay in the their current state because of a higher probability even though the optimal state value of that state itself is lower than it's neighboring nodes.

The second discrepancy is the confusion of the correct policy, leading the agent to go toward the wrong or even opposite direction. This occurs because these states are influenced by local rewards that we extracted through IRL algorithm, which can accumulate rapidly in the incorrect direction and overshadow the potential maximum reward, making the agent to be confused about the correct direction.




In the above figure, the presence of non-zero rewards in the state-space of Question 21 introduces complexity into the agent's decision-making process. Unlike in Question 7, where most rewards are zero and the agent can easily discern optimal paths, the varied rewards in Question 21 may lead the agent astray. Immediate rewards that seem attractive may lure the agent off the grid, especially if they overpower the long-term rewards associated with optimal paths. Consequently, the agent may deviate from the intended trajectory, potentially leading to suboptimal outcomes.

In the above figure, certain actions point towards each other, indicating the presence of local optima in the policy plot. This phenomenon creates a deadlock condition where the agent becomes trapped in a loop, oscillating between two states indefinitely. The root cause of this situation lies in the gradual changes in immediate rewards among neighboring states. If the rewards between two neighboring states are similar and non-zero, the agent may struggle to determine the best course of action. With a low exploration probability, the agent may become stuck in a cycle of moving back and forth between these states, unable to progress towards the optimal solution.

To fix the first discrepancy, we modify and set up a border constraint for all states at the edges or corner, besides state 99, in the value iteration algorithm by replacing the value to negative infinity so that the action, which causes the agent moving off the grid, is basically eliminated and could not be able to choose as the optimal policy at the end. With such modification, two incorrect arrows / policies have been fixed and the maximum accuracy increase from 96% to 98%.