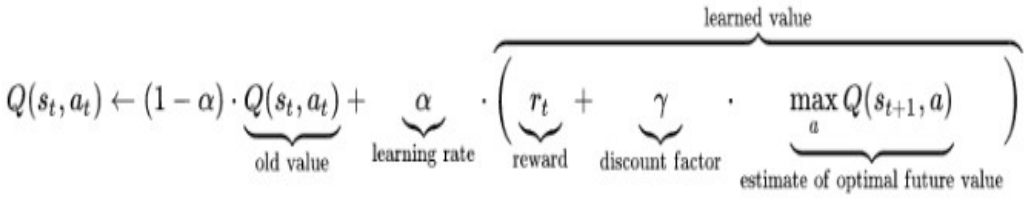
Reinforcement Learning

Define a learning rule (equation) for the Q-function <u>and</u> describe how it works. (Theory, see lectures/classes)

We have used Bellman Equation as a learning rule.



This is called the Bellman optimality equation. It states that, for any state-action pair (s,a) at time t, the expected return from starting in state s, selecting action a and following the optimal policy thereafter is going to be the expected reward we get from taking action a in state s, which is Rt+1, plus the maximum expected discounted return that can be achieved from any possible next state-action pair (s′,a′).

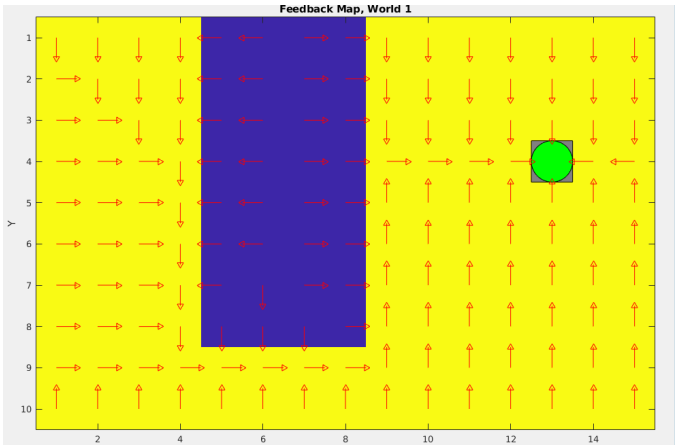
3. **Briefly describe your implementation, especially how you hinder the robot from exiting through the borders of a world.**
When robot hits the wall, we give that action large negative feedback(-1000). Upon exploiting, the max (optimal move) is always bigger than -1000 so the invalid move is not picked by the robot.
For all valid moves we update the q table according to the learning rate and the feedback received for that state using the equation stated below. This worked well for the first three world, where the actions of the robot were not random.

Update equation is:
qnew(s,a)=(1−α) * q(s,a) + α * (Rt+1 + γ * max′ q(s′,a′))

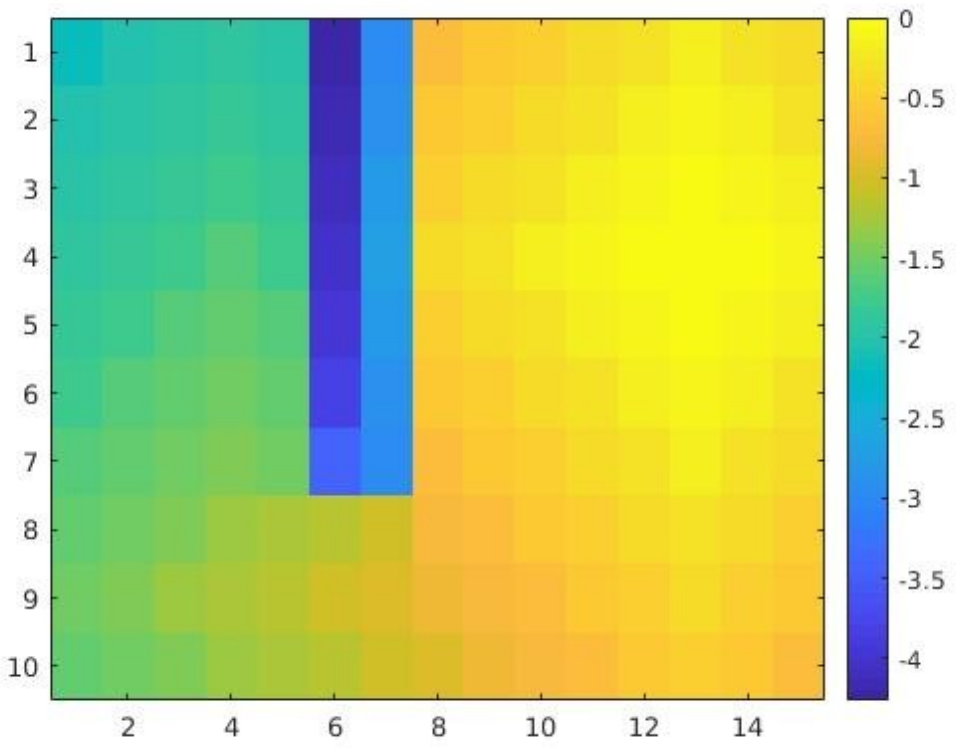
The α in this equation is the learning rate, that controls how much do we have to remember from the old q-value and how much weight we must give to the new learned value. Its value is in the range (0-1).

4. **Describe World 1. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.**
World 1 is simplest of all the worlds in which the goal is to reach the destination(green point) covering the shortest distance. The world contains a patch of water which have negative reward so robot has to avoid that patch.
For this world, parameters were:
   - Number of iteration or episode : 3000
   - Exploration rate: 1
   - Maximum Exploration rate: 1
   - Minimum Exploration rate: 0.2

- Exploration Decay: 0.001
- Alpha: 0.1
- Gamma/ Discount rate: 0.9
- Max Steps in each episode: 1000



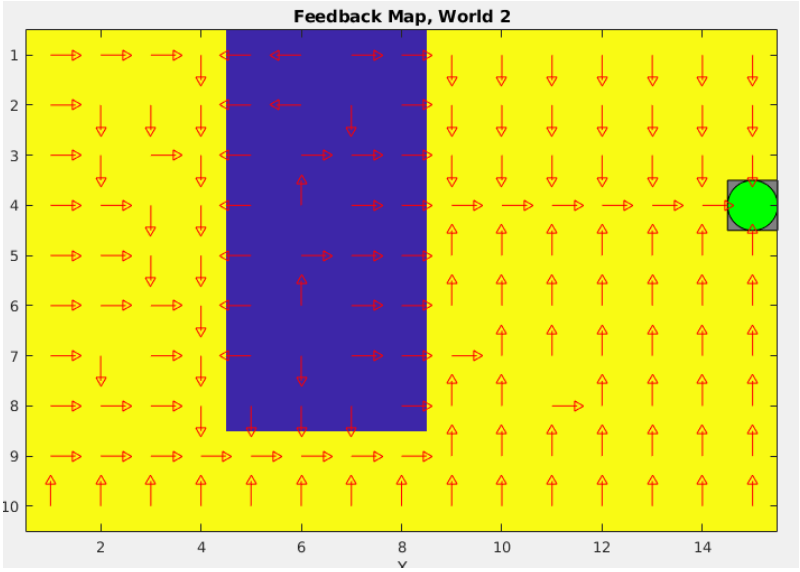
The V-function plot is:



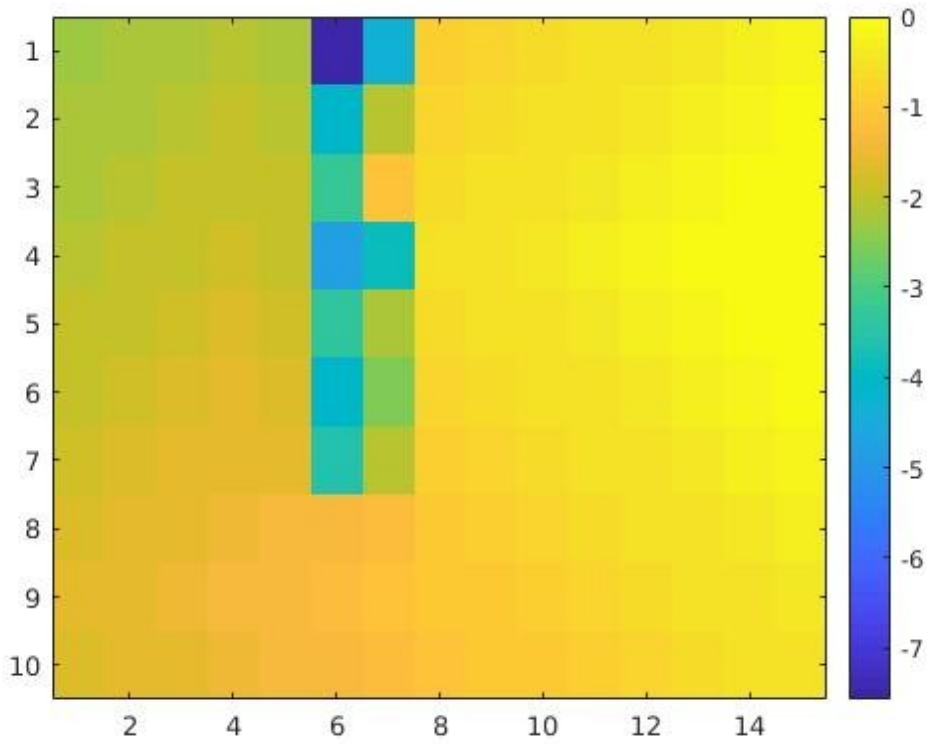
5. **Describe World 2. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.**

   World 2 is bit more hard to learn as the destination point lies near the boundary and there is a chance (20% probability) that world1 is developed. And even when world1 Is generated the robot assumes that the patch could be there ans hence chooses the best policy generated by avoiding the patch in both worlds to be safe.
   For this world, parameters were:

- Number of iteration or episode : 4000
- Exploration rate: 1
- Maximum Exploration rate: 1
- Minimum Exploration rate: 0.2
- Exploration Decay: 0.001
- Alpha: 0.2
- Gamma/ Discount rate: 0.9
- Max Steps in each episode: 1000



The V-function plot is:



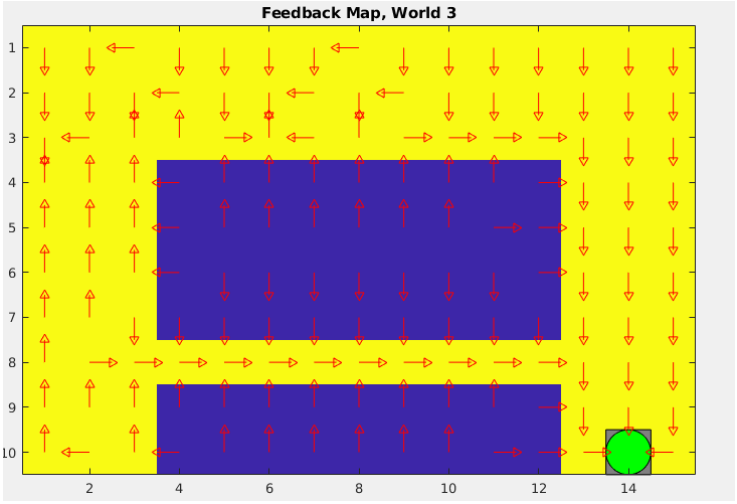
6. **Describe World 3. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.**

World 3 is more advance than previous two worlds and had more area covering with water and there is no randomness in its generation. The shortest path is the passage between the water bodies which is quite hard to learn as compared to previous worlds.
We used same parameters:

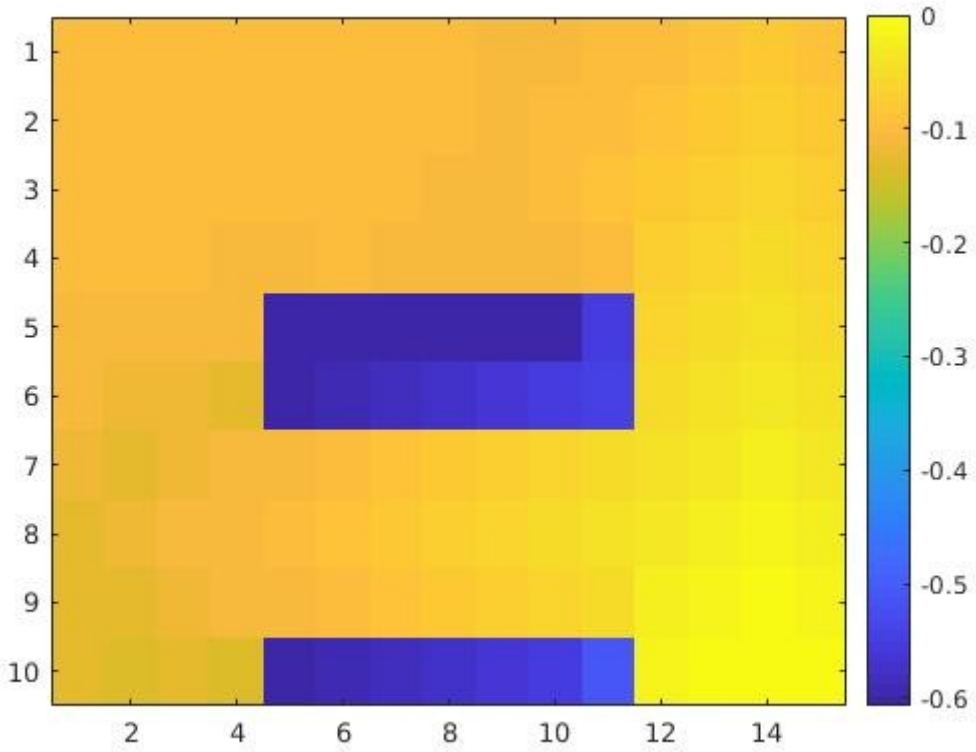
For this world, parameters were:

- Number of iteration or episode : 4000
- Exploration rate: 1
- Maximum Exploration rate: 1
- Minimum Exploration rate: 0.2
- Exploration Decay: 0.001
- Alpha: 0.2
- Gamma/ Discount rate: 0.9
- Max Steps in each episode: 1000

Policy plot is as follows:



The V-function plot is:



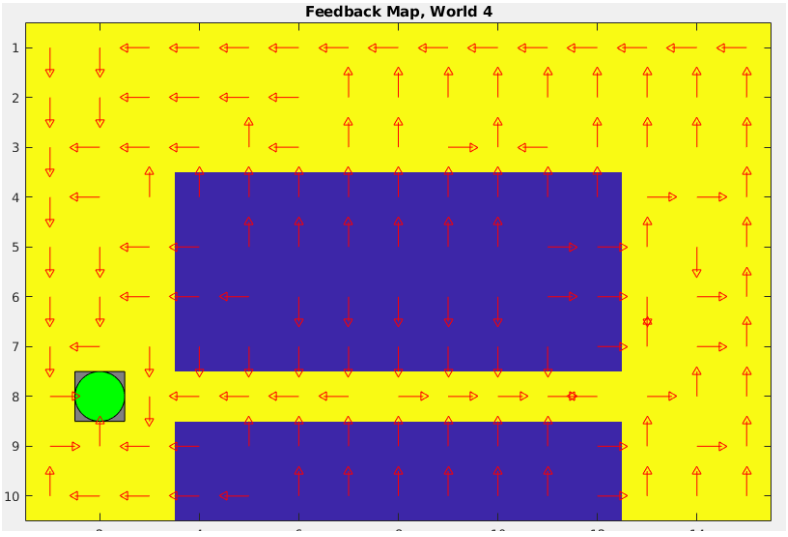
7. **Describe World 4. What is the goal of the reinforcement learning in this world? How is this world different from world 3, and why can this be solved using reinforcement learning? What parameters did you use to solve this world? Plot the policy and the V-function.**

   World 4 looked like world 3 but instead, the moves that robot makes were randomized with 30% probability which, sometimes, lead to inaccurate feedback for valid moves and caused the robot to get stuck. So we had to train the robot iis such a way that it could ovvercome reach the goal even when the moves were randomized with some probability. For this, we manually, added large negative feedback in the Q-table for invalud moves.

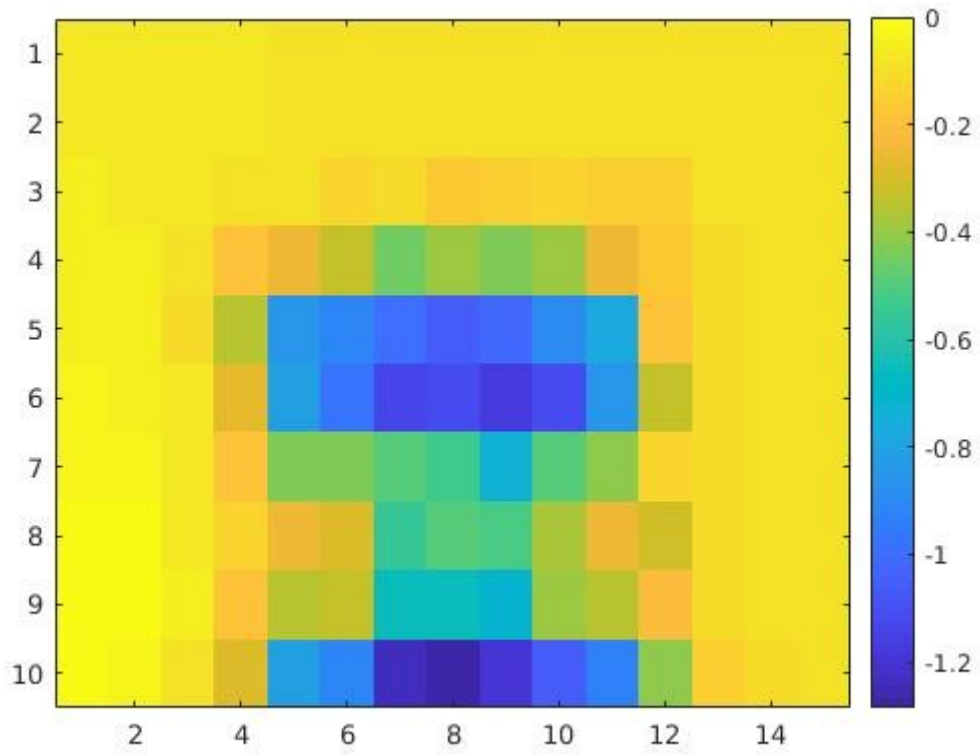
   We used the same parameter:

   - Number of iteration or episode : 4000
   - Exploration rate: 1
   - Maximum Exploration rate: 1
   - Minimum Exploration rate: 0.2
   - Exploration Decay: 0.001
   - Alpha: 0.2
   - Gamma/ Discount rate: 0.9
   - Max Steps in each episode: 1000

   Policy plot is:



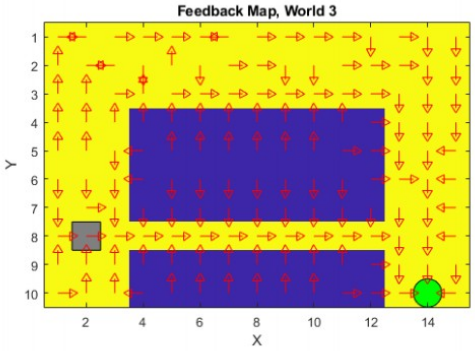
The V-function plot is:



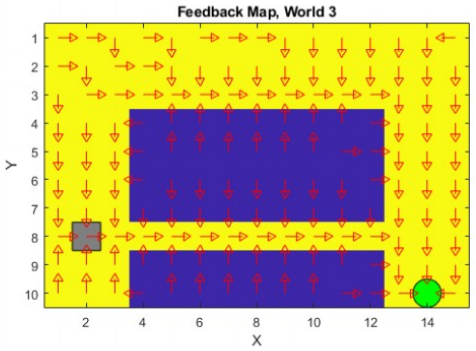
8. **Explain how the learning rate α influences the policy and V-function in each world. Use figures to make your point.**

   The learning rate is basically how fast we want our robot to update the information in the Q-table. It is the way we decide how agent will preserve the information of the previous state and how will it pas that information to the new state in Q-table.

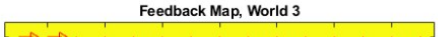
   With increase in learning rate, more and more parallel lines started to appear going towards the goal.On increasing the learning rate for the 4th world, where the actions are random, the optimal policy starts getting messy. It needs the previous knowledge to find an optimal policy through the randomness in the world. So, higher learning rates are good for worlds with no randomness, but for worlds with randomness the learning rate needs to be small in order to tackle the randomness.

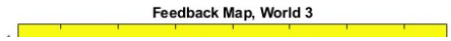


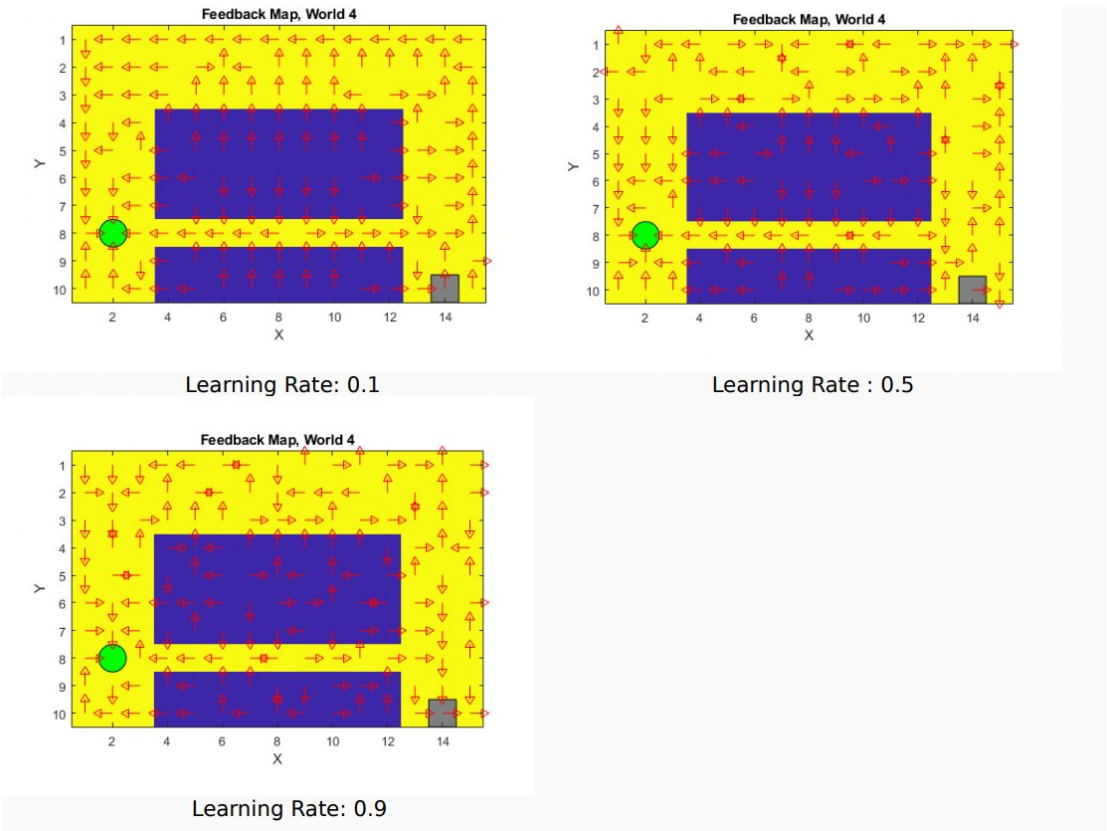
Learning Rate: 0.1



Learning Rate : 0.5

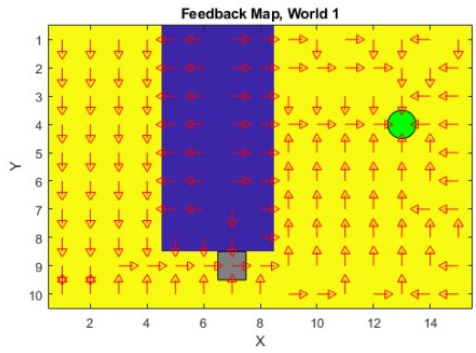




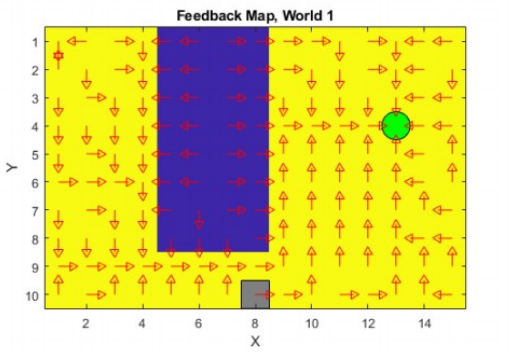


Alpha 2: World 4 with different learning rates

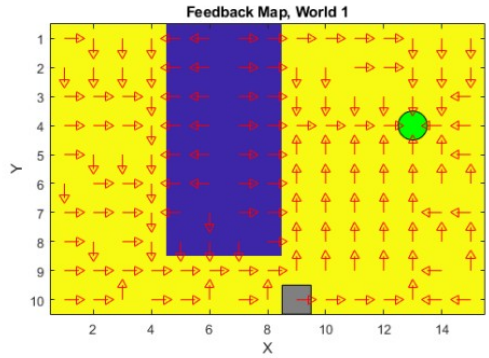
9. **Explain how the discount factor γ influences the policy and V-function in each world. Use figures to make your point.**
   It is the discount factor, which represents the difference in future and present rewards. This models the fact that future rewards are worth less than immediate rewards. Mathematically, the discount factor needs to be set less than 0 for the algorithm to converge.



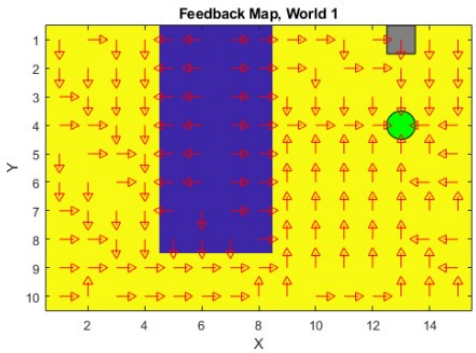
Discount: 0.1



Discount: 0.5



Discount: 0.99



Discount: 1.0

10. **Explain how the exploration rate ε influences the policy and V-function in each world. Use figures to make your point. Did you use any strategy for changing ε during training?**
    One of the most challenging tasks in reinforcement learning is that of balancing the ratio between exploration and exploitation. Too much exploration yields a lower accumulated reward, while too much exploitation can lead to the agent being stuck in a local optimum.
    Want to explore more in the beginning (large e) of the training phase and less towards the end.

    We updated **ε as follows:**
    ε = min**ε + (max**ε **-min**ε **)*exp(-decay**ε ***epoch);**

11. **What would happen if we instead of reinforcement learning were to use Dijkstra's cheapest path finding algorithm in the "Suddenly Irritating blob" world? What about in the static "Irritating blob" world?**

    Since all of our rewards are negative it would be fairly straightforward to restate the problem as one of minimizing costs instead of maximizing rewards, interpret the

world as a graph with costs on transitions, and apply Dijkstra to solve it. The question was aimed more at considering how this straightforward solution only applies to the static worlds that we used, and not so much to the ones with randomness built into them
because there is no suitable procedure to deal situation where the weigths can suddenly change or where a different step is taken than the intended one.

Dijkstra's algorithm would require a defined goal state, and we don't have that before training. So, the algorithm would not be able to converge. Dijkstra's algorithm is just the opposite, it requires no training, but take a long time when testing, and is bad as the world keeps getting larger.

12. **Can you think of any application where reinforcement learning could be of practical use? A hint is to use the Internet.**

    Yes there exists different applications that utilize reinforcement learning.

    Robotics.RL can be used for high-dimensional control problems as well as various industrial applications. Google, for example, has reportedly cut its energy consumption by about 50% after implementing Deep Mind's technologies. There are innovative startups in the space (Bonsai, etc.) that are propagating deep reinforcement learning for efficient machine and equipment tuning.

    Text mining. The researchers from Salesforce, a renowned cloud computing company, used RL along with an advanced contextual text generation model to develop a system that's able to produce highly readable summaries of long texts. According to them, one can train their algorithm on different types of material (news articles, blogs, etc.).

    Trade execution. Major companies in the financial industry have been using ML algorithms to enhance trading and equity for a while and some of them, such as JPMorgan, have already thrown their hats into the RL ring too. The company announced in 2017 that it would start using a robot for trading execution of large orders. Their model, trained on billions of historic transactions, would allow to the execute trading procedure promptly, at optimal prices, and offload huge stakes without creating market swings.

13. **(Optional) Try your implementation in the other available worlds 5-12. Does it work in all of them, or did you encounter any problems, and in that case how would you solve them?**