Autonomous Robots Lab Report 5
Reinforcement Learning
(Q-Learning Algorithm)

Raabid Hussain

May 19, 2016

1 Objective

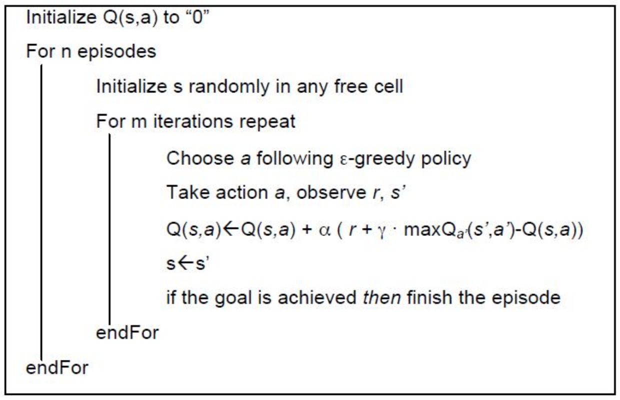
The objective of this lab work was to implement the Q-learning algorithm for finding optimum path planning strategy using MATLAB. We were provided with a discretized map of the environment with the goal position. Given the map, we were to determine an optimum approach to go from any position in the map to the goal position using reinforcement learning techniques. The results were graphically represented to provide a better insight of the proposed solution.

2 Introduction

Reinforcement learning is one of the important ideas in machine learning strategies. It is inspired by behavior psychology. It aims at maximizing some reward function by taking different actions. Owing to the nature of the algorithm, it has been employed in different areas like controls, robotics, swarm intelligence, statistics, economics, game theory etc. The primary use of such techniques is to find the characteristics of the optimal solution by utilizing dynamic programming techniques.The main advantage of the reinforcement learning techniques is that they do not require prior knowledge about the environment.The algorithm does not necessarily present the correct output, rather there is always a trade-off between exploration of the map and its exploitation.

Reinforcement learning techniques have also been utilized in determining the optimum path planning strategy. A typical algorithm is the Q-learning algorithm with involves finding the path by optimizing am action value function. The algorithm involves extensive repetitions before reaching the optimum policy. The algorithm works by first assigning a reward value for reaching each cell in the map. Typically the reward depends on how close the cell is to the goal position. Next a random value is assigned to each cell as its action value function (Q).This is the value that needs to be optimized. A separate copy of this action value is made to represent each movement the robot can make separately.

A typical pseudo-code for Q-learning strategy as used in path planning is given in figure 1. Each iteration known as an episode starts at a random cell in the environment. Than for certain number of times a valid action is taken. This action is chosen by an epsilon greedy policy by taking into account the results of the previous iterations. The action value for the current cell is updated using the reward obtained after taking that action. For the next action iteration, the current cell is changed to be the cell that would have been reached after the particular action has been taken. If the goal position has been reached or a maximum number of action iterations have been reached, the episode stops and the next one is started.



```
Initialize Q(s,a) to "0"
For n episodes
    Initialize s randomly in any free cell
    For m iterations repeat
        Choose $a$ following $\varepsilon$-greedy policy
        Take action $a$, observe $r$, $s'$
        $Q(s,a) \leftarrow Q(s,a) + \alpha\,(r + \gamma \cdot \max Q_{a'}(s',a') - Q(s,a))$
        $s \leftarrow s'$
        if the goal is achieved *then* finish the episode
    endFor
endFor
```

Figure 1: Pseudo-code for the Q-learning algorithm.

3 Implementation of Q-Learning Algorithm

MATLAB was used to implement the Q-learning algorithm for path planning. A discretized map of the environment was provided to us along with the goal position. An example environment is displayed in figure 2 where the black regions represent free space and the white regions represent the obstacles. We had to implement the Q-learning strategy in order to obtain an optimum policy such that if the robot starts at any position in the environment, it would be able to reach the goal position following that optimum strategy.

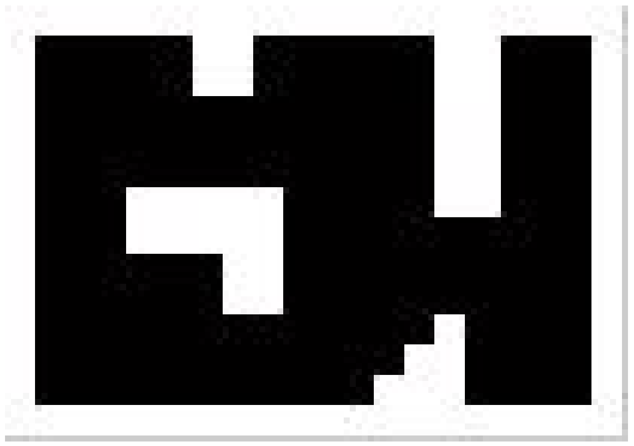
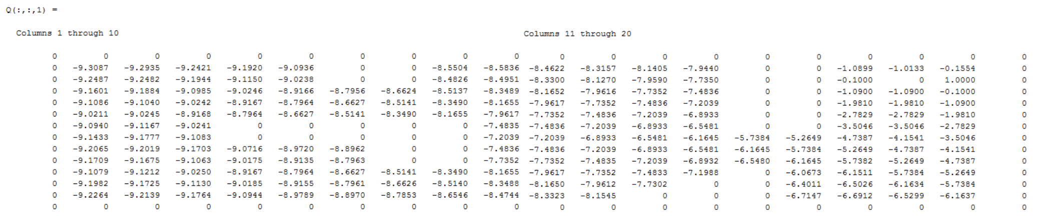


Figure 2: A sample map of the environment used to test the algorithm.

A matrix equivalent to the size map was initialized to zero. In total four such matrices were built, each representing the Q-action value for each valid action.The action chosen were left, up, right and down. For simplicity, the reward of '+1' was assigned for the goal position, whereas the reward for all other cells was kept to be -1. Next the loop mentioned in figure 1 was used to obtain the optimum strategy. Each episode was randomly initialized at a cell in the map. A greedy action was taken. The reward assigned to the new cell in the map and the best action value for the cell (among all the actions) were used to update the Q action value for the current cell depending on a discount factor and a learning rate. The new cell is taken to be the current cell in the next iteration. If the new cell is an obstacles, the current cell is again used as we are trying to imitate the actual behavior in a practical environment.

The Q action values for each action are shown in figures 3, 4, 5 and 6. The values represent a measure of the reward for taking a particular action if that cell/position is reached.

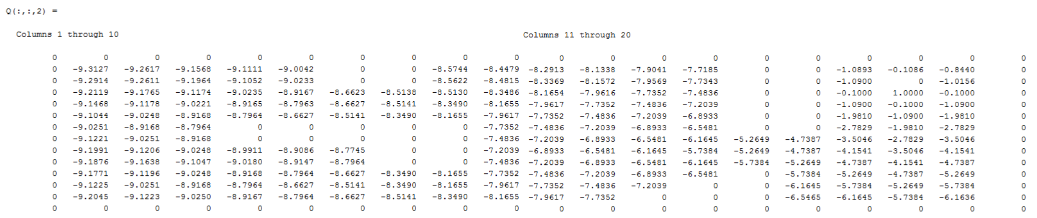


```
Q(:,:,1) =

  Columns 1 through 10                                                      Columns 11 through 20

     0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
     0   -9.3087   -9.2935   -9.2421   -9.1920   -9.0936         0         0   -8.5504   -8.5836   -8.4622   -8.3157   -8.1405   -7.9440         0         0   -1.0899   -1.0133   -0.1554         0
     0   -9.2487   -9.2482   -9.1944   -9.1150   -9.0238         0         0   -8.4826   -8.4951   -8.3300   -8.1270   -7.9590   -7.7350         0         0   -0.1000         0    1.0000         0
     0   -9.1601   -9.1884   -9.0985   -9.0246   -8.9166   -8.7956   -8.6624   -8.5137   -8.3489   -8.1652   -7.9616   -7.7352   -7.4836         0         0   -1.0900   -1.0900   -0.1000         0
     0   -9.1086   -9.1040   -9.0242   -8.9167   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039         0         0   -1.9810   -1.9810   -1.0900         0
     0   -9.0211   -9.0245   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933         0         0   -2.7829   -2.7829   -1.9810         0
     0   -9.0940   -9.1167   -9.0241         0         0         0         0         0   -7.4835   -7.4836   -7.2039   -6.8933   -6.5481         0         0   -3.5046   -3.5046   -2.7829         0
     0   -9.1433   -9.1777   -9.1083         0         0         0         0         0   -7.2039   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -3.5046         0
     0   -9.2065   -9.2019   -9.1703   -9.0716   -8.9720   -8.8962         0         0   -7.4836   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541         0
     0   -9.1709   -9.1675   -9.1063   -9.0175   -8.9135   -8.7963         0         0   -7.7352   -7.7352   -7.4835   -7.2039   -6.8932   -6.5480   -6.1645   -5.7382   -5.2649   -4.7387         0
     0   -9.1079   -9.1212   -9.0250   -8.9167   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4833   -7.1988         0   -6.0673   -6.1511   -5.7384   -5.2649         0
     0   -9.1982   -9.1725   -9.1130   -9.0185   -8.9155   -8.7961   -8.6626   -8.5140   -8.3488   -8.1650   -7.9612   -7.7302         0         0   -6.4011   -6.5026   -6.1634   -5.7384         0
     0   -9.2264   -9.2139   -9.1764   -9.0944   -8.9789   -8.8970   -8.7853   -8.6546   -8.4744   -8.3323   -8.1545         0         0         0   -6.7147   -6.6912   -6.5299   -6.1637         0
     0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
```

Figure 3: The Q action values for each cell if the action taken is to move left.

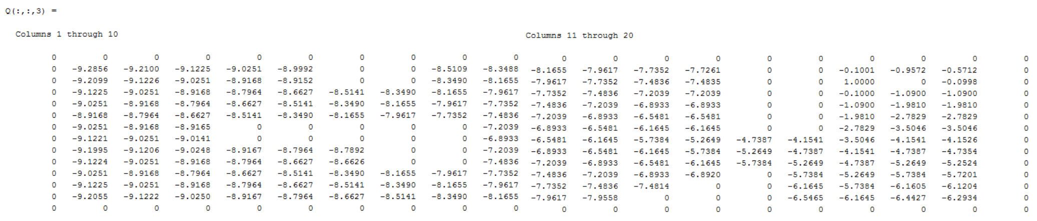


```
Q(:,:,2) =

  Columns 1 through 10                                                      Columns 11 through 20

     0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
     0   -9.3127   -9.2617   -9.1568   -9.1111   -9.0042         0         0   -8.5744   -8.4479   -8.2913   -8.1338   -7.9041   -7.7185         0         0   -1.0893   -0.1086   -0.8440         0
     0   -9.2914   -9.2611   -9.1964   -9.1052   -9.0233         0         0   -8.5622   -8.4815   -8.3369   -8.1572   -7.9569   -7.7343         0         0   -1.0900         0   -1.0156         0
     0   -9.2119   -9.1765   -9.1174   -9.0235   -8.9167   -8.6623   -8.5138   -8.5130   -8.3486   -8.1654   -7.9616   -7.7352   -7.4836         0         0   -0.1000    1.0000   -0.1000         0
     0   -9.1468   -9.1178   -9.0221   -8.9165   -8.7963   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039         0         0   -1.0900   -0.1000   -1.0900         0
     0   -9.1044   -9.0248   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933         0         0   -1.9810   -1.0900   -1.9810         0
     0   -9.0251   -8.9168   -8.7964         0         0         0         0         0   -7.7352   -7.4836   -7.2039   -6.8933   -6.5481         0         0   -2.7829   -1.9810   -2.7829         0
     0   -9.1221   -9.0251   -8.9168         0         0         0         0         0   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645   -5.2649   -4.7387   -3.5046   -2.7829   -3.5046         0
     0   -9.1991   -9.1206   -9.0248   -8.9911   -8.9086   -8.7745         0         0   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -3.5046   -4.1541         0
     0   -9.1876   -9.1638   -9.1047   -9.0180   -8.9147   -8.7964         0         0   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -4.7387         0
     0   -9.1771   -9.1196   -9.0248   -8.9168   -8.7964   -8.6627   -8.3490   -8.1655   -7.7352   -7.4836   -7.2039   -6.8933   -6.5481         0   -5.7384   -5.2649   -4.7387   -5.2649         0
     0   -9.1225   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039         0         0   -6.1645   -5.7384   -5.2649   -5.7384         0
     0   -9.2045   -9.1223   -9.0250   -8.9167   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352         0         0         0   -6.5465   -6.1645   -5.7384   -6.1636         0
     0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
```

Figure 4: The Q action values for each cell if the action taken is to move upwards.

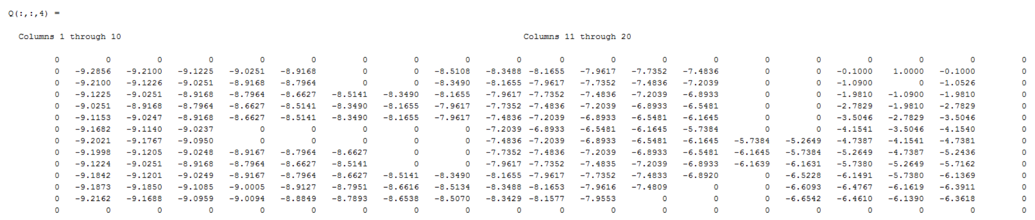


```
Q(:,:,3) =

  Columns 1 through 10                                                      Columns 11 through 20

     0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
     0   -9.2856   -9.2100   -9.1225   -9.0251   -8.9992         0         0   -8.5109   -8.3488   -8.1655   -7.9617   -7.7352   -7.7261         0         0   -0.1001   -0.9572   -0.5712         0
     0   -9.2099   -9.1226   -9.0251   -8.9168   -8.9152         0         0   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.4835         0         0    1.0000         0   -0.0998         0
     0   -9.1225   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -7.2039         0         0   -0.1000   -1.0900   -1.0900         0
     0   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933   -6.8933         0         0   -1.0900   -1.9810   -1.9810         0
     0   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933   -6.5481   -6.5481         0         0   -1.9810   -2.7829   -2.7829         0
     0   -9.0251   -8.9168   -8.9165         0         0         0         0         0   -7.2039   -6.8933   -6.5481   -6.1645   -6.1645         0         0   -2.7829   -3.5046   -3.5046         0
     0   -9.1221   -9.0251   -9.0141         0         0         0         0         0   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -3.5046   -4.1541   -4.1526         0
     0   -9.1995   -9.1206   -9.0248   -8.9167   -8.7964   -8.7892         0         0   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -4.7387   -4.7354         0
     0   -9.1224   -9.0251   -8.9168   -8.7964   -8.6627   -8.6626         0         0   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -5.2649   -5.2524         0
     0   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933   -6.8920         0   -5.7384   -5.2649   -5.7384   -5.7201         0
     0   -9.1225   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.4814         0         0   -6.1645   -5.7384   -6.1605   -6.1204         0
     0   -9.2055   -9.1222   -9.0250   -8.9167   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.9558         0         0         0   -6.5465   -6.1645   -6.4427   -6.2934         0
     0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
```

Figure 5: The Q action values for each cell if the action taken is to move right.



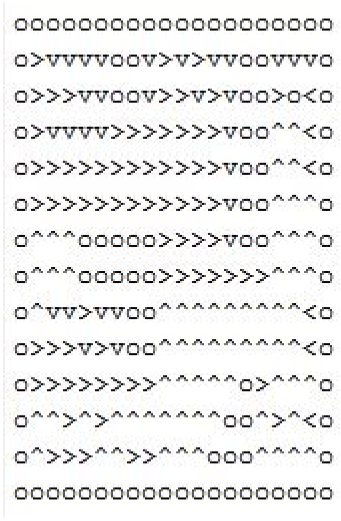
```
Q(:,:,4) =

  Columns 1 through 10                                                                              Columns 11 through 20

         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
         0   -9.2856   -9.2100   -9.1225   -9.0251   -8.9168         0         0   -8.5108   -8.3488   -8.1655   -7.9617   -7.7352   -7.4836         0         0   -0.1000    1.0000   -0.1000         0
         0   -9.2100   -9.1226   -9.0251   -8.9168   -8.7964         0         0   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039         0         0   -1.0900         0   -1.0526         0
         0   -9.1225   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933         0         0   -1.9810   -1.0900   -1.9810         0
         0   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933   -6.5481         0         0   -2.7829   -1.9810   -2.7829         0
         0   -9.1153   -9.0247   -8.9168   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645         0         0   -3.5046   -2.7829   -3.5046         0
         0   -9.1682   -9.1140   -9.0237         0         0         0         0         0   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384         0         0   -4.1541   -3.5046   -4.1540         0
         0   -9.2021   -9.1767   -9.0950         0         0         0         0         0   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -4.7381         0
         0   -9.1998   -9.1205   -9.0248   -8.9167   -8.7964   -8.6627         0         0   -7.7352   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -5.2436         0
         0   -9.1224   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141         0         0   -7.9617   -7.7352   -7.4835   -7.2039   -6.8933   -6.1639   -6.1631   -5.7380   -5.2649   -5.7162         0
         0   -9.1842   -9.1201   -9.0249   -8.9167   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4833   -6.8920         0   -6.5228   -6.1491   -5.7380   -6.1369         0
         0   -9.1873   -9.1850   -9.1085   -9.0005   -8.9127   -8.7951   -8.6616   -8.5134   -8.3488   -8.1653   -7.9616   -7.4809         0         0   -6.6093   -6.4767   -6.1619   -6.3911         0
         0   -9.2162   -9.1688   -9.0959   -9.0094   -8.8849   -8.7893   -8.6538   -8.5070   -8.3429   -8.1577   -7.9553         0         0         0   -6.6542   -6.4610   -6.1390   -6.3618         0
         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
```

Figure 6: The Q action values for each cell if the action taken is to move downwards.

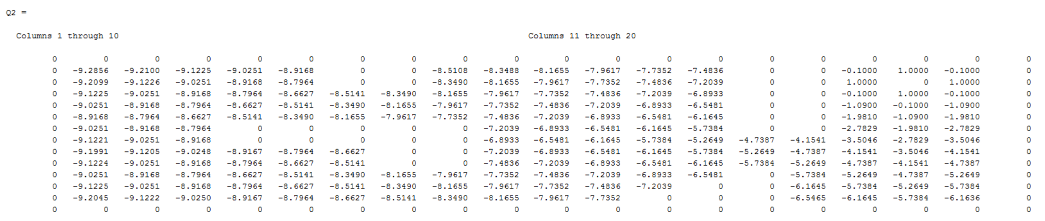
After obtaining the Q action values for each cell and action, it was time to extract the best policy for path planning. For this the action with the best Q value was chosen for each cell. The best policy for our sample environment is displayed in figure 7. The actions are represented by arrows. Since, we were limited to submit only 1 m-file with the code for the algorithm, a better representation using images for directions was not possible. Also, the arrows have been compressed while showing as the algorithm has to be working on larger environments too and by this it will be better for visualizing the optimum policy for larger maps.



```
oooooooooooooooooooo
o>vvvvoov>v>vvoovvvo
o>>>vvoov>>v>voo>o<o
o>vvvv>>>>>>>voo^^<o
o>>>>>>>>>>>>voo^^<o
o>>>>>>>>>>>>voo^^^o
o^^^ooooo>>>>voo^^^o
o^^^ooooo>>>>>>>^^^o
o^vv>vvoo^^^^^^^^^<o
o>>>v>voo^^^^^^^^^<o
o>>>>>>>>^^^^^o>^^^o
o^^>^>^^^^^^^oo^>^<o
o^>>>^^>>^^^ooo^^^^o
oooooooooooooooooooo
```

Figure 7: The optimum policy for the sample environment. Arrows point towards the action that should be taken if the robot reaches that particular cell.

The Q values for the best action was chosen to be the output of the algorithm. The best values for each cell are displayed in figure 8 while a graphical representation is shown in figure 9. As can be observed from the two figures, the action value is higher as we move towards the goal position as the value depends on the reward.



```
Q2 =

  Columns 1 through 10                                                                  Columns 11 through 20

         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
         0   -9.2856   -9.2100   -9.1225   -9.0251   -8.9168         0         0   -8.5108   -8.3488   -8.1655   -7.9617   -7.7352   -7.4836         0         0   -0.1000    1.0000   -0.1000         0
         0   -9.2099   -9.1226   -9.0251   -8.9168   -8.7964         0         0   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039         0         0    1.0000         0    1.0000         0
         0   -9.1225   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933         0         0   -0.1000    1.0000   -0.1000         0
         0   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933   -6.5481         0         0   -1.0900   -0.1000   -1.0900         0
         0   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645         0         0   -1.9810   -1.0900   -1.9810         0
         0   -9.0251   -8.9168   -8.7964         0         0         0         0         0   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384         0         0   -2.7829   -1.9810   -2.7829         0
         0   -9.1221   -9.0251   -8.9168         0         0         0         0         0   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -3.5046   -2.7829   -3.5046         0
         0   -9.1991   -9.1205   -9.0248   -8.9167   -8.7964   -8.6627         0         0   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -3.5046   -4.1541         0
         0   -9.1224   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141         0         0   -7.4836   -7.2039   -6.8933   -6.5481   -6.1645   -5.7384   -5.2649   -4.7387   -4.1541   -4.7387         0
         0   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039   -6.8933   -6.5481         0   -5.7384   -5.2649   -4.7387   -5.2649         0
         0   -9.1225   -9.0251   -8.9168   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352   -7.4836   -7.2039         0         0   -6.1645   -5.7384   -5.2649   -5.7384         0
         0   -9.2045   -9.1222   -9.0250   -8.9167   -8.7964   -8.6627   -8.5141   -8.3490   -8.1655   -7.9617   -7.7352         0         0         0   -6.5465   -6.1645   -5.7384   -6.1636         0
         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
```

Figure 8: The optimum Q action values for each cell using the best action for each cell.

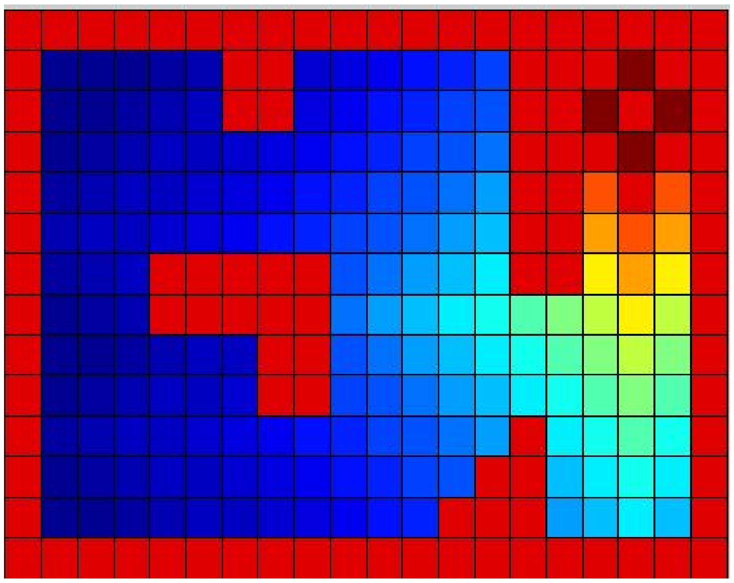


Figure 9: Graphical representation of the best Q action values.

After implementing the algorithm, the effectiveness for the algorithm on a particular environment was found. The effectiveness is a measure of the reward for a given set of episodes. In our case the effectiveness was measured after every 200 episodes. For computing the effectiveness, only the best action was taken at each cell. The reward for each iteration was summed and divided by the number of episodes for a given effectiveness measure. For the next episodes the effectiveness was dropped back to zero and the reward accumulated. A graph for all the accumulated rewards is shown in figure 10. As expected, the graph starts at around -45 to -50 and increases steadily as the number of iterations increase. This is a reflection of the fact that when the environment has been more explored, the algorithm adapts better to the input environment thus raising the effectiveness measure.

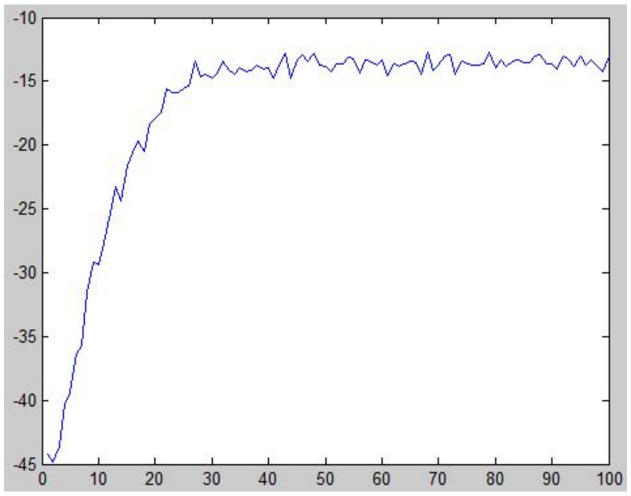


Figure 10: Effectiveness graph for the sample environment (effectiveness vs (x 100) iterations).

4 Conclusion

In this lab assignment, the Q-learning algorithm was successfully implemented using MATLAB. Provided a discretized map of the environment and the goal position, the algorithm determines the best action to be taken for each cell. This allows the robot to reach the goal position using an optimum path irrespective of the starting location of the robot/movement. The effectiveness of the algorithm was also determined. The algorithm was tested on different environments however, only the results for the sample environment is shown in the report. All the tests were successful.