HW2

Kin Chang (013783848)

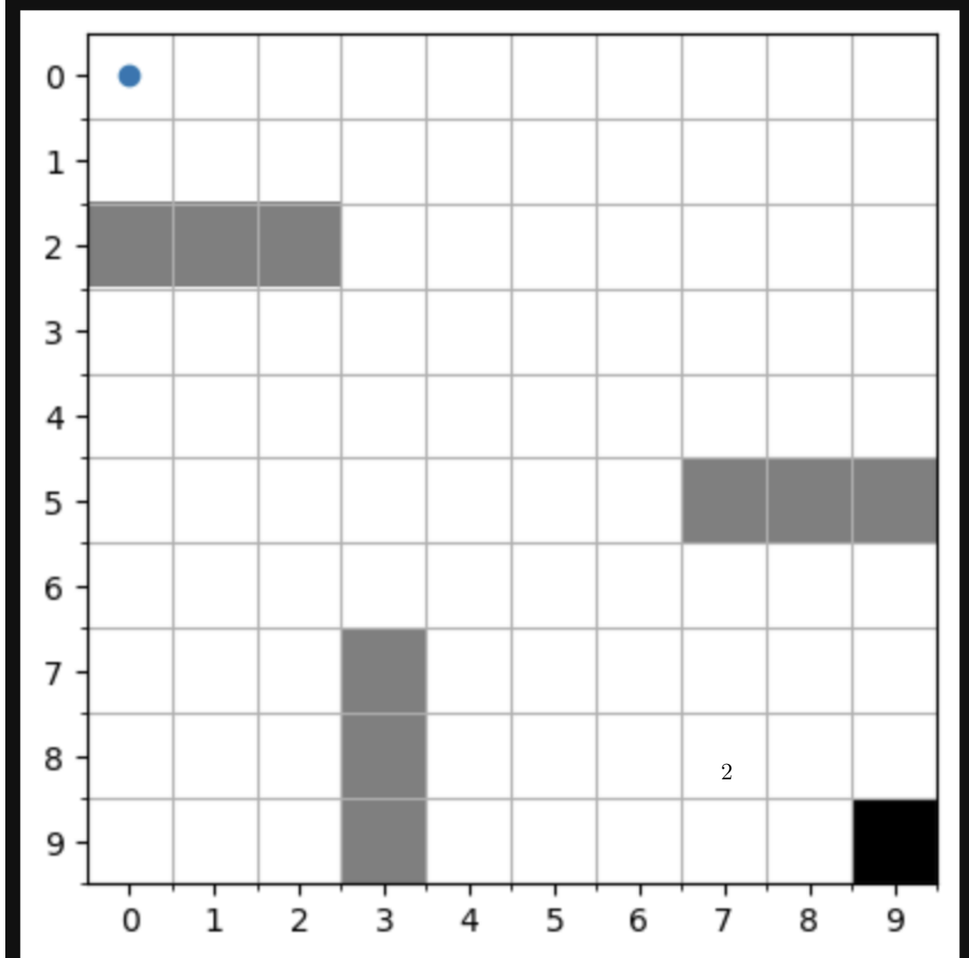
<2023-04-07 Fri>

Task1

- Made a Maze class for easy access to dS, dA, actions, goal, obstacles .etc that are common to functions like visualization, valid state, policy evaluation
- Pytorch tensors are used to represent our state-transformation function, policy functions, value functions, reward functions and more
- Visualization make use of matplotlib, such as scatter(), imshow(), etc
- For more detail, pleaserefere the maze class in the jupyter notebook
- A screenshot of the maze visualization is shown below

```python
# Configurations
obst1 = [(x, 2) for x in range(3)]
obst2 = [(3, y) for y in range(9, 6, -1)]
obst3 = [(x, 5) for x in range(9, 6, -1)]
dS = 10
#Up, Right, Left, Down, Stay
actions = [(0, -1), (1, 0), (-1, 0), (0, 1), (0, 0)]
dA = len(actions)
goal = (9,9)
obstacles = obst1 + obst2 + obst3
maze = Maze(dS, dA, actions, goal, obstacles)
```

```python
maze.visualize((0,0))
```

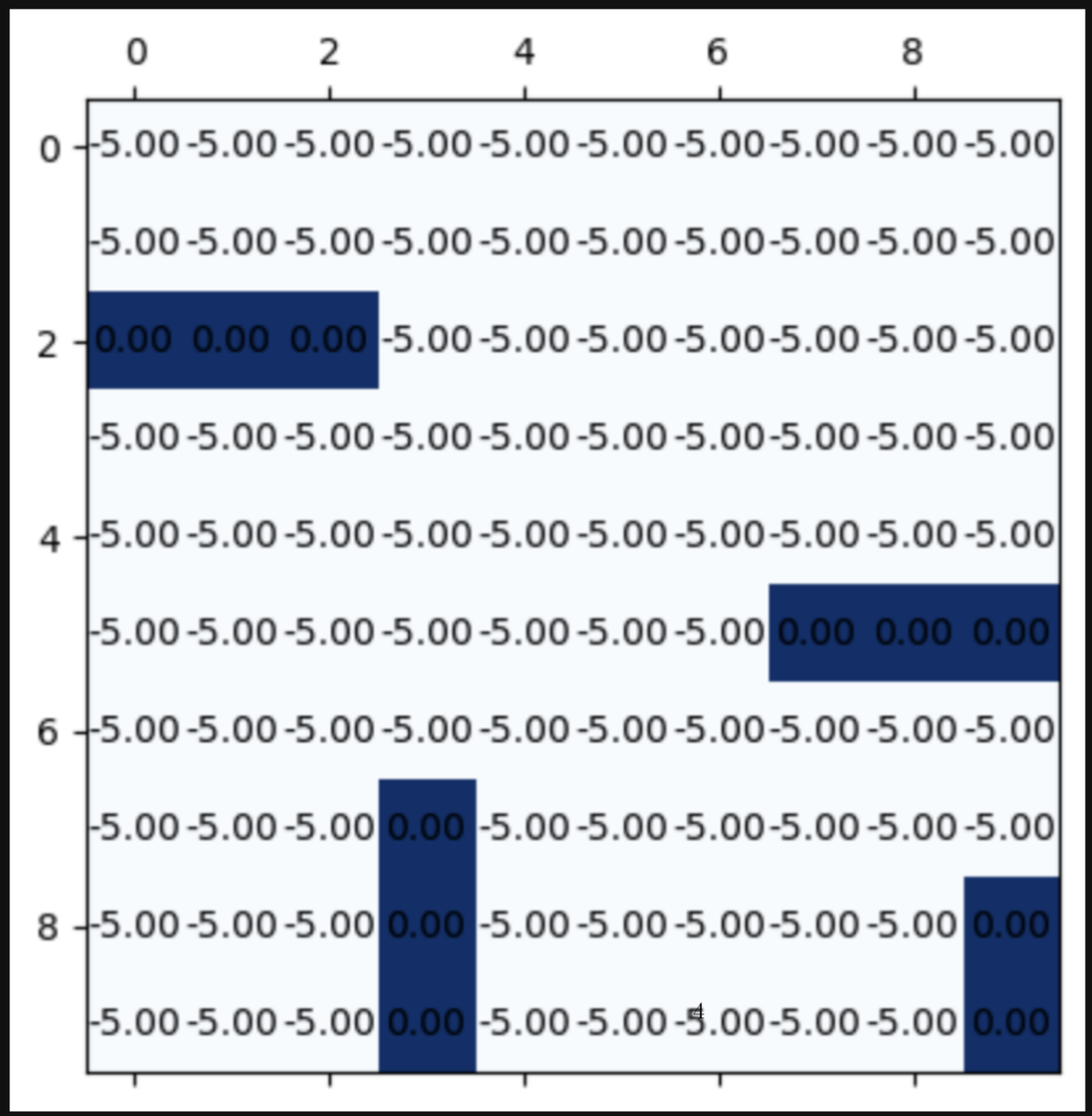


Task2

- Implemented get random policy, where policy(s) do one random action with probability of 1 (deterministic)
- Instead of (y, x), actions are modified to be in the format of (x, y), where x is from left to right, and y is from top to bottom
  - Up, Right, Left, Down, Stay = [(0, -1), (1, 0), (-1, 0), (0, 1), (0, 0)]
  - the (x, y) format will also be the default coordinate in the entire Maze class
- Policy evaluation is implemented according to the updating rule. With terminating condition controlled by the hyperparameter threshold.
- A sample run with gamma=0.8 and threshold=0.001 is shown below. Note that at s=(9,8), the evaluation is 0, because it happens that the random generated policy((9,8)) is to go DOWN, which happens to be optimal. In contrast at s=(8,9), the evaluation is -5, because the random generated policy((8,9)) is to go UP, which is a bad policy.

Task 2

```
pi1 = maze.getRandomPolicy()
v1 = maze.policy_evaluation(pi1, gamma=0.8, threshold=0.001)
maze.visualize_matrix(v1)
```




```
print("Policy at s=(9,8) is: {}".format(actionToWords(pi1[9,8])))
print("Policy at s=(8,9) is: {}".format(actionToWords(pi1[8,9])))
```

Task3

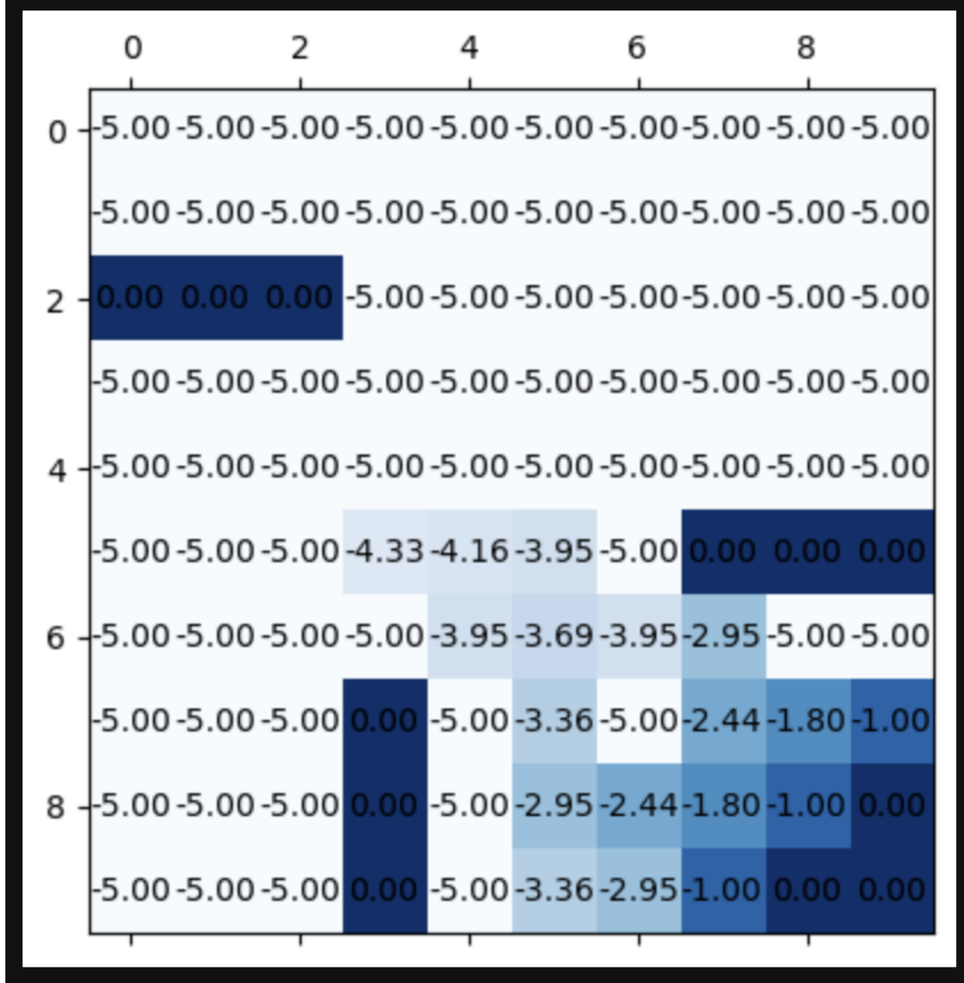
- Implemented set optimal policy with radius as a parameter. The logic is essentially for all state within the radius from the goal, set all the last row to go right, last column to go down, and all others to go left or down with probability of 50 respectively

- A sample run with the policy evaluation with this improved policy is shown below. Note that all states within 2 radius from goal now has better value than that of task2.

- All other states that have an action that can reach a state within 2 radius of the goal is also improved! For example, policy at s=(5,9) is to go RIGHT, which gets closer to the goal (radius 2); policy at s=(7,6) is to go DOWN, which also gets closer to the goal (radius 2).

Task 3 ¶

```
pi2 = maze.getRandomPolicy()
maze.setOptimalPolicyFromGoal(pi2)
v2 = maze.policy_evaluation(pi2, gamma=0.8, threshold=0.001)
maze.visualize_matrix(v2)
```

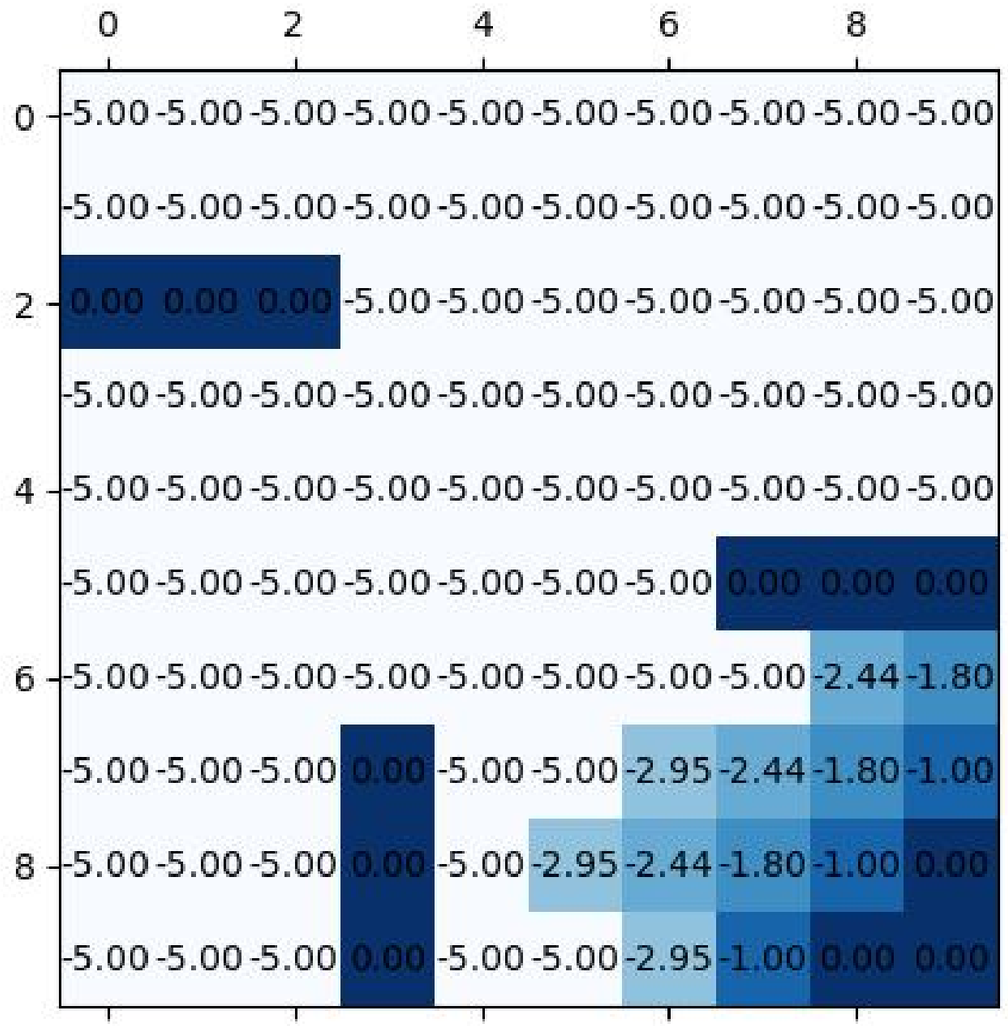


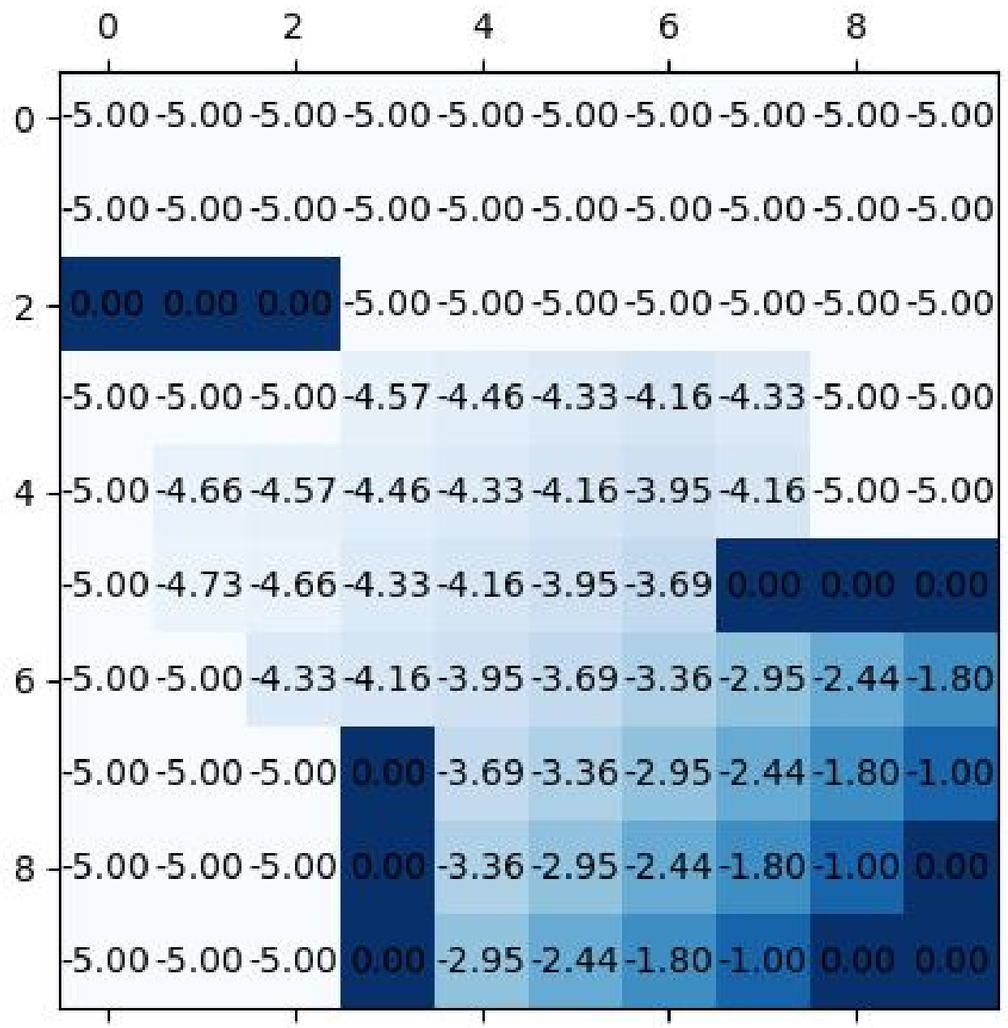

```
print("Policy at s=(5,9) is: {}".format(actionToWords(pi2[5,9])))
print("Policy at s=(7,6) is: {}".format(actionToWords(pi2[7,6])))
```

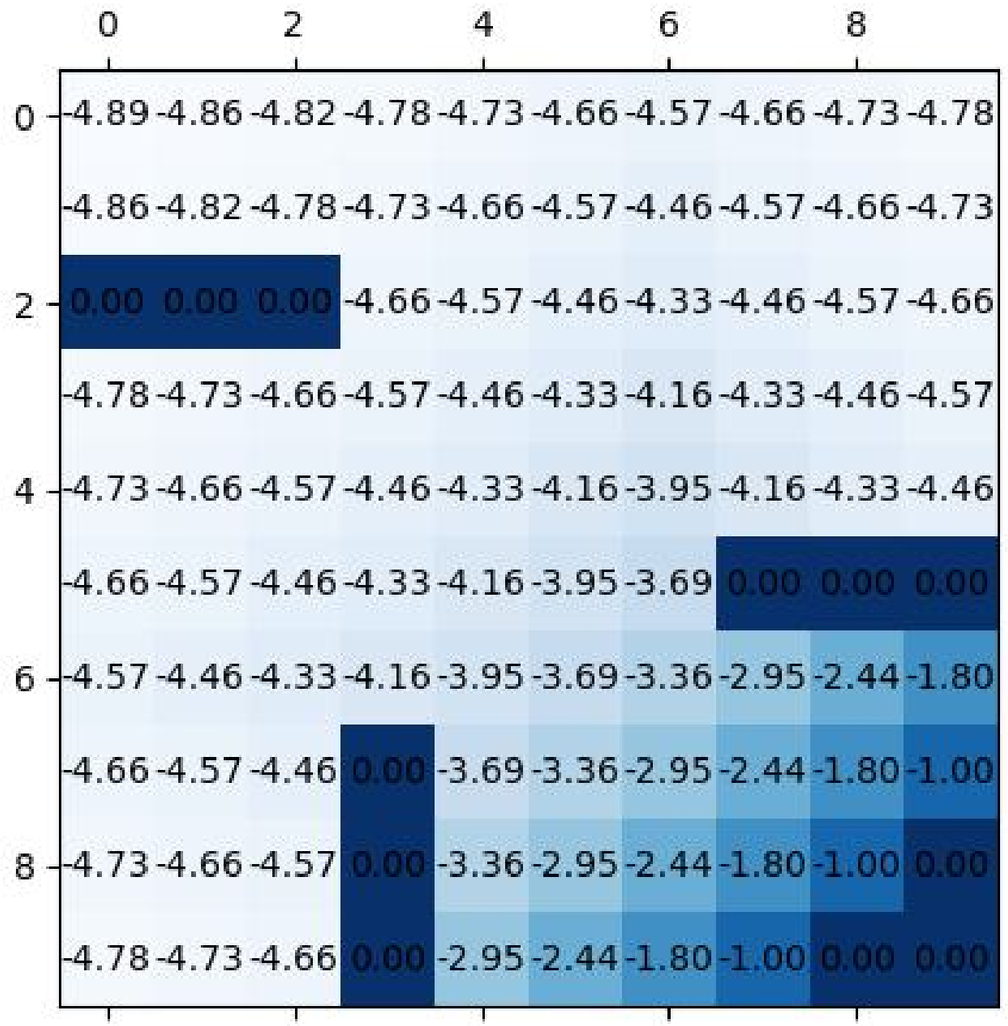
```
Policy at s=(5,9) is: ['RIGHT']
Policy at s=(7,6) is: ['DOWN']
```

Task4

- Implemented the policy improvement function, as well as the policy iteration that combines evaluation and improvement
- Note that in policy improvement, the policy(s) probability isn't multiplied in the sum, because we are exploring other actions to see if it's better
- For gamma=0.8 and threshold=0.01, the iteration takes 11 iteration to complete. The progression at i=2, 5, and final are shown below


0 2 4 6 8
0 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00
-5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00
2 0.00 0.00 0.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00
-5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00
4 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00
-5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 0.00 0.00 0.00
6 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -2.44 -1.80
-5.00 -5.00 -5.00 0.00 -5.00 -5.00 -2.95 -2.44 -1.80 -1.00
8 -5.00 -5.00 -5.00 0.00 -5.00 -2.95 -2.44 -1.80 -1.00 0.00
-5.00 -5.00 -5.00 0.00 -5.00 -5.00 -2.95 -1.00 0.00 0.00


0 2 4 6 8
0 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00
-5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00
2 0.00 0.00 0.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00 -5.00
-5.00 -5.00 -5.00 -4.57 -4.46 -4.33 -4.16 -4.33 -5.00 -5.00
4 -5.00 -4.66 -4.57 -4.46 -4.33 -4.16 -3.95 -4.16 -5.00 -5.00
-5.00 -4.73 -4.66 -4.33 -4.16 -3.95 -3.69 0.00 0.00 0.00
6 -5.00 -5.00 -4.33 -4.16 -3.95 -3.69 -3.36 -2.95 -2.44 -1.80
-5.00 -5.00 -5.00 0.00 -3.69 -3.36 -2.95 -2.44 -1.80 -1.00
8 -5.00 -5.00 -5.00 0.00 -3.36 -2.95 -2.44 -1.80 -1.00 0.00
-5.00 -5.00 -5.00 0.00 -2.95 -2.44 -1.80 -1.00 0.00 0.00


0 2 4 6 8
0 -4.89 -4.86 -4.82 -4.78 -4.73 -4.66 -4.57 -4.66 -4.73 -4.78
-4.86 -4.82 -4.78 -4.73 -4.66 -4.57 -4.46 -4.57 -4.66 -4.73
2 0.00 0.00 0.00 -4.66 -4.57 -4.46 -4.33 -4.46 -4.57 -4.66
-4.78 -4.73 -4.66 -4.57 -4.46 -4.33 -4.16 -4.33 -4.46 -4.57
4 -4.73 -4.66 -4.57 -4.46 -4.33 -4.16 -3.95 -4.16 -4.33 -4.46
-4.66 -4.57 -4.46 -4.33 -4.16 -3.95 -3.69 0.00 0.00 0.00
6 -4.57 -4.46 -4.33 -4.16 -3.95 -3.69 -3.36 -2.95 -2.44 -1.80
-4.66 -4.57 -4.46 0.00 -3.69 -3.36 -2.95 -2.44 -1.80 -1.00
8 -4.73 -4.66 -4.57 0.00 -3.36 -2.95 -2.44 -1.80 -1.00 0.00
-4.78 -4.73 -4.66 0.00 -2.95 -2.44 -1.80 -1.00 0.00 0.00