Isabelle Chalhoub - u0678302
Karla Kraiss - u0830999
CS4300
November 30, 2017

**A8 Lab Report**

## 1. Introduction

The purpose of this project is to create algorithms for Utility and Policy generation using the Markov Decision Problem. We are calculating utilities of cells on our Wumpus board using our scoring system as our reward policy and the actions:

A = {UP, LEFT, DOWN, RIGHT}

By calculating expected utilities for each state in our wumpus board:

```
0 0 0 G
0 0 P 0
0 0 W 0
0 0 P 0
```

We can then create a policy based on the expected utilities that will provide an agent with a path to the good terminal state (gold) and will avoid bad terminal states like pits or the wumpus.

- How much does the effectiveness of policy iteration change based on the number of iterations allowed?
- How does the effectiveness differ when given a varying range of error thresholds (gamma)?

## 2. Method

**CS4300_MDP_policy_iteration:**

Based on the pseudo-code provided in the text page 657 for a modified policy iteration algorithm. It starts by using an initial policy with associated expected utilities. Then the algorithm calculates a new policy using one step look ahead for the best expected utilities. The algorithm continues this pattern of improving the policy steadily until it reaches an “equilibrium” where the policy is improved as much as it can be.

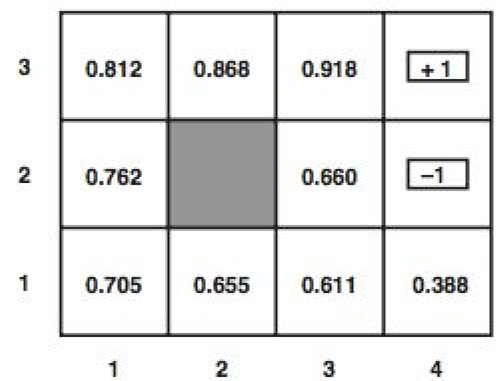
**CS4300_policy_eval:**

Calculates the expected utilities given a policy and an initial set of utilities. We chose to implement this the iterative way matching the equation given in the text instead of the system of linear equations as discussed in class.

## 3. Verification of Program

To verify, our MDP algorithm, we ran it using the framework for a 4x3 board as explained in the textbook in figure 17.3 to see if we got matching expected utilities. We used the reward at -0.04 just like the book. Here are our results for this framework:

```
>> [S,A,R,P,U,Ut, policy] = CS4300_run_3x4();
>> U

U =

    0.7053
    0.6553
    0.6114
    0.3879
    0.7616
         0
    0.6603
   -1.0000
    0.8116
    0.8678
    0.9178
    1.0000
```




Our results (above and below)

Book results ^

| | | | |
|---|---|---|---|
| .8116 | .8678 | .9178 | 1 |
| .7616 | 0 | .6603 | -1 |
| .7053 | .6553 | .6114 | .3879 |

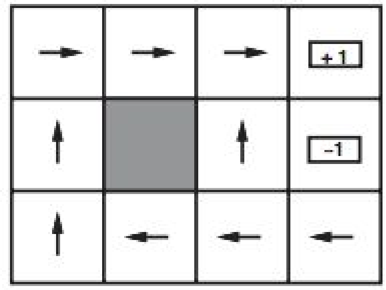
Once our MDP algorithm was confirmed, we checked the final policy produced against the book's:

```
>> [S,A,R,P,U,Ut, policy] = CS4300_run_3x4();
>> policy

policy =

     1     2     2     2     1     1     1     1     4     4     4     1
```

1-4 maps to [UP, LEFT, DOWN, RIGHT]

Below is a more readable version compared with the book's board on the right. We changed our policies in states 6, 8, and 12 to represent the terminal states.

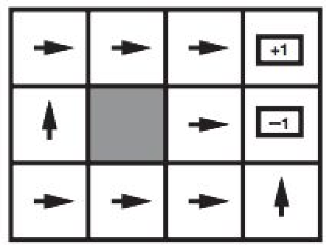
| | | | |
|---|---|---|---|
| 4 - RIGHT | 4 - RIGHT | 4 - RIGHT | +1 |
| 1 - UP | | 1 - UP | -1 |
| 1 - UP | 2 - LEFT | 2 - LEFT | 2 - LEFT |




Next we tested the same board with different starting policies (only readable versions). Our boards are on the left, the books boards are on the right for all following examples.

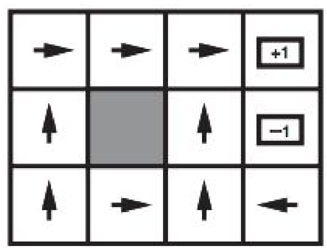
**Reward = -1.8**

| | | | |
|---|---|---|---|
| 4 - RIGHT | 4 - RIGHT | 4 - RIGHT | +1 |
| 1 - UP | | 4 - RIGHT | -1 |
| 4 - RIGHT | 4 - RIGHT | 4 - RIGHT | 1 - UP |



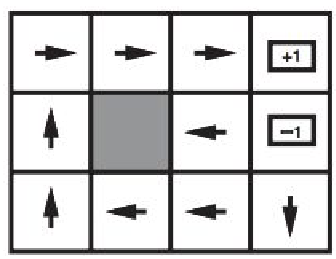

**Reward = -0.0850**

| | | | |
|---|---|---|---|
| 4 - RIGHT | 4 - RIGHT | 4 - RIGHT | +1 |
| 1 - UP | | 1 - UP | -1 |
| 1 - UP | 4 - RIGHT | 1 - UP | 2 - LEFT |



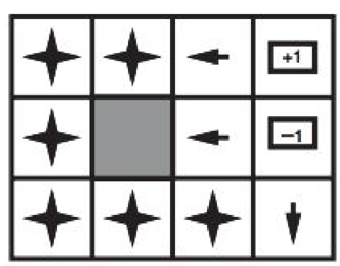

**Reward = -0.02**

| | | | |
|---|---|---|---|
| 4 - RIGHT | 4 - RIGHT | 4 - RIGHT | +1 |
| 1 - UP | | 2 - LEFT | -1 |
| 1 - UP | 2 - LEFT | 2 - LEFT | 3 - DOWN |

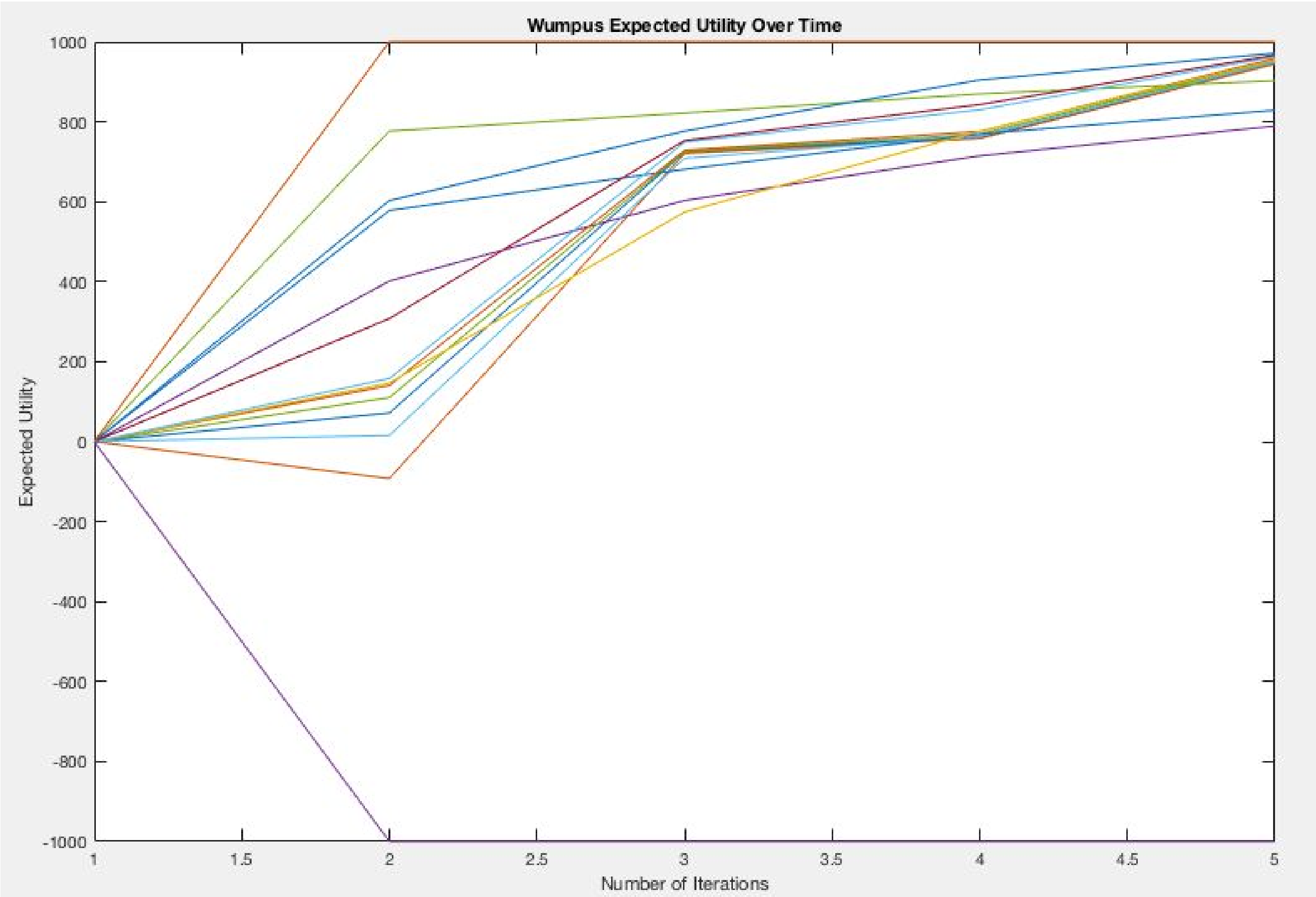


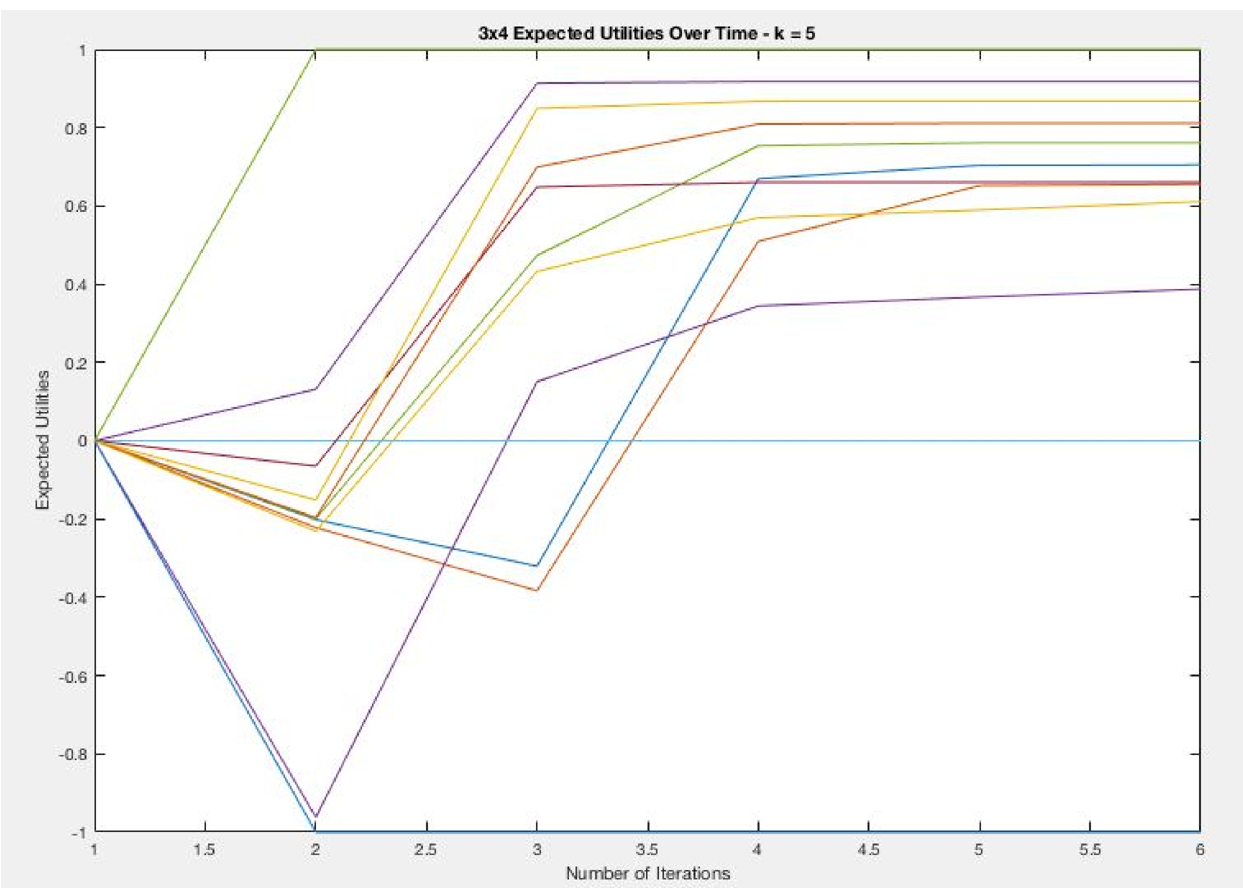

**Reward = 2**

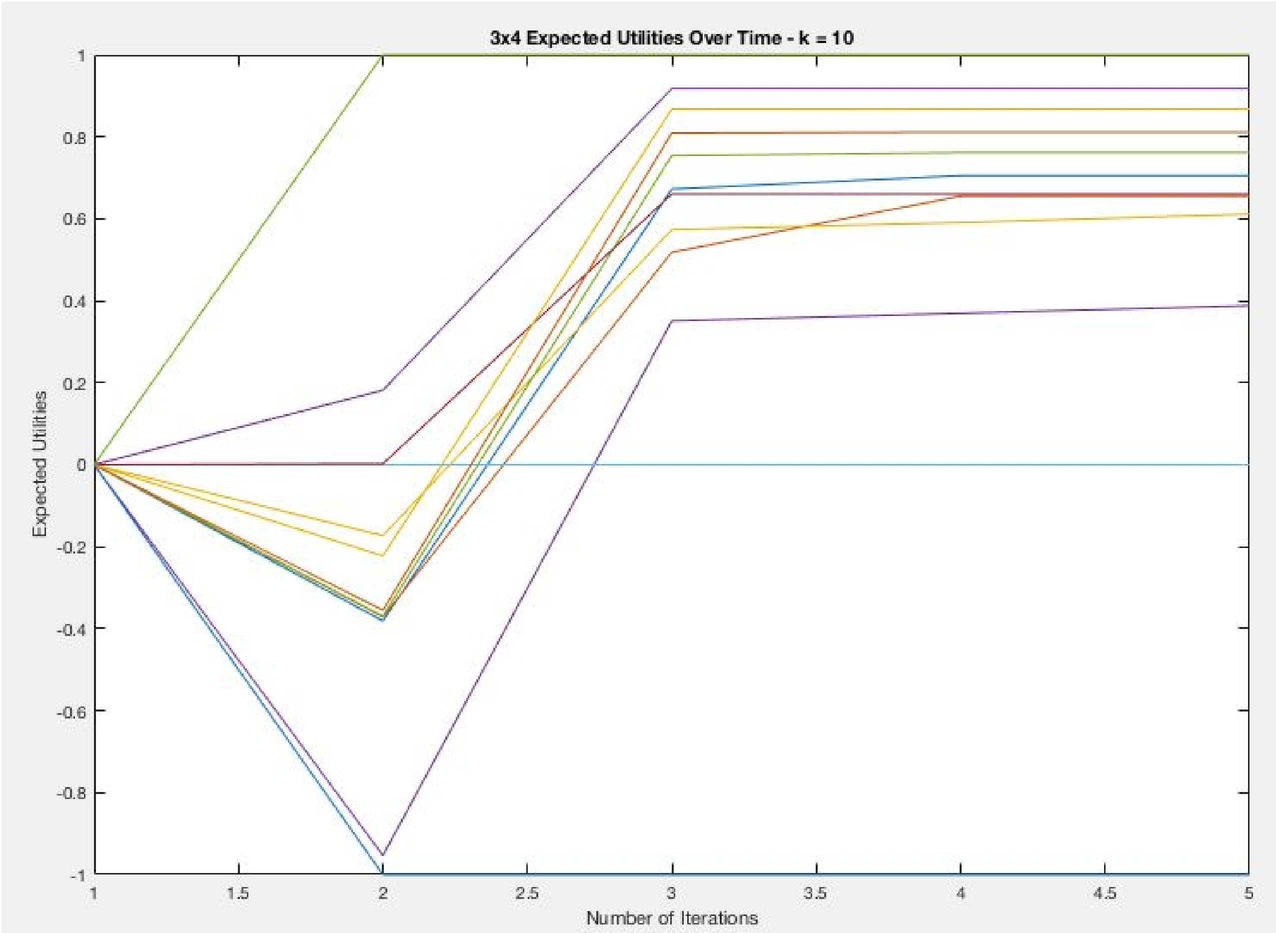
| | | | |
|---|---|---|---|
| 1 - UP | 1 - UP | 2 - LEFT | +1 |
| 2 - LEFT | | 2 - LEFT | -1 |
| 2 - LEFT | 1 - UP | 2 - LEFT | 3 - DOWN |

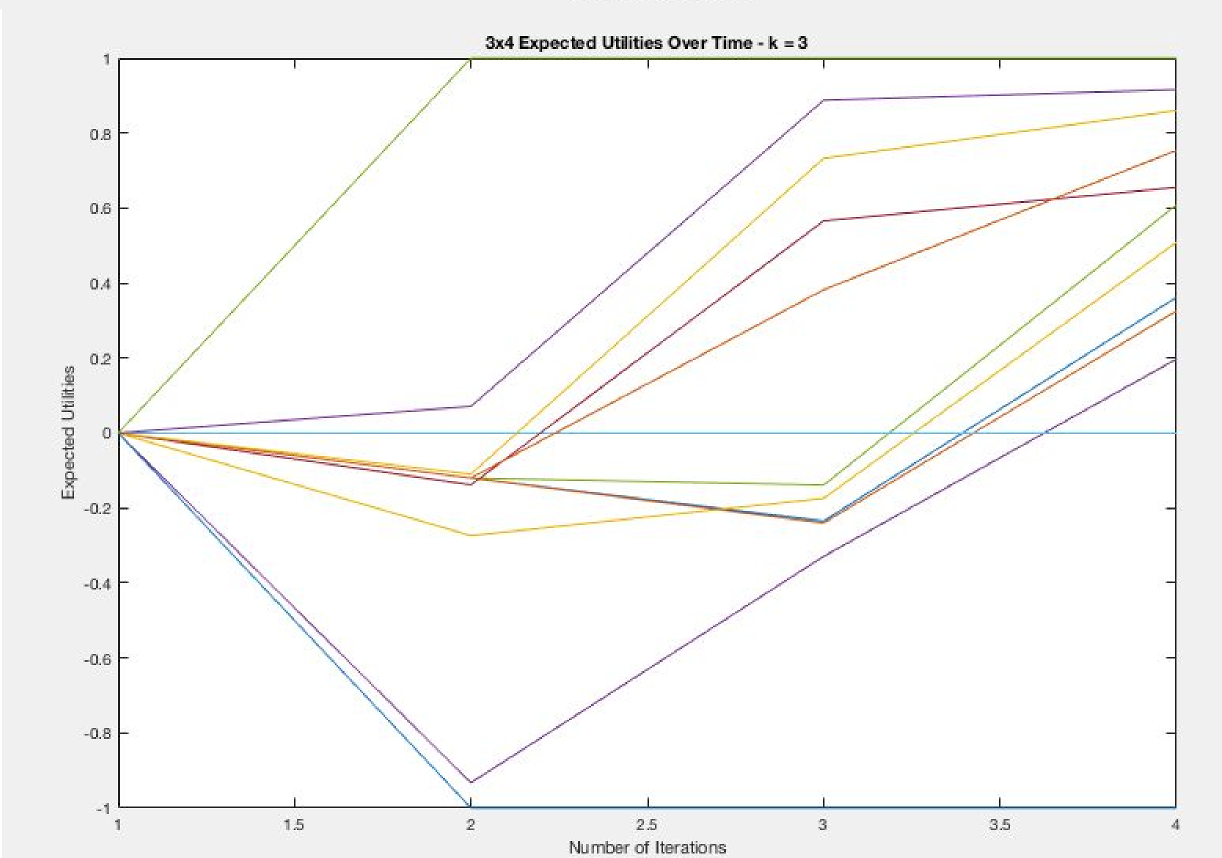


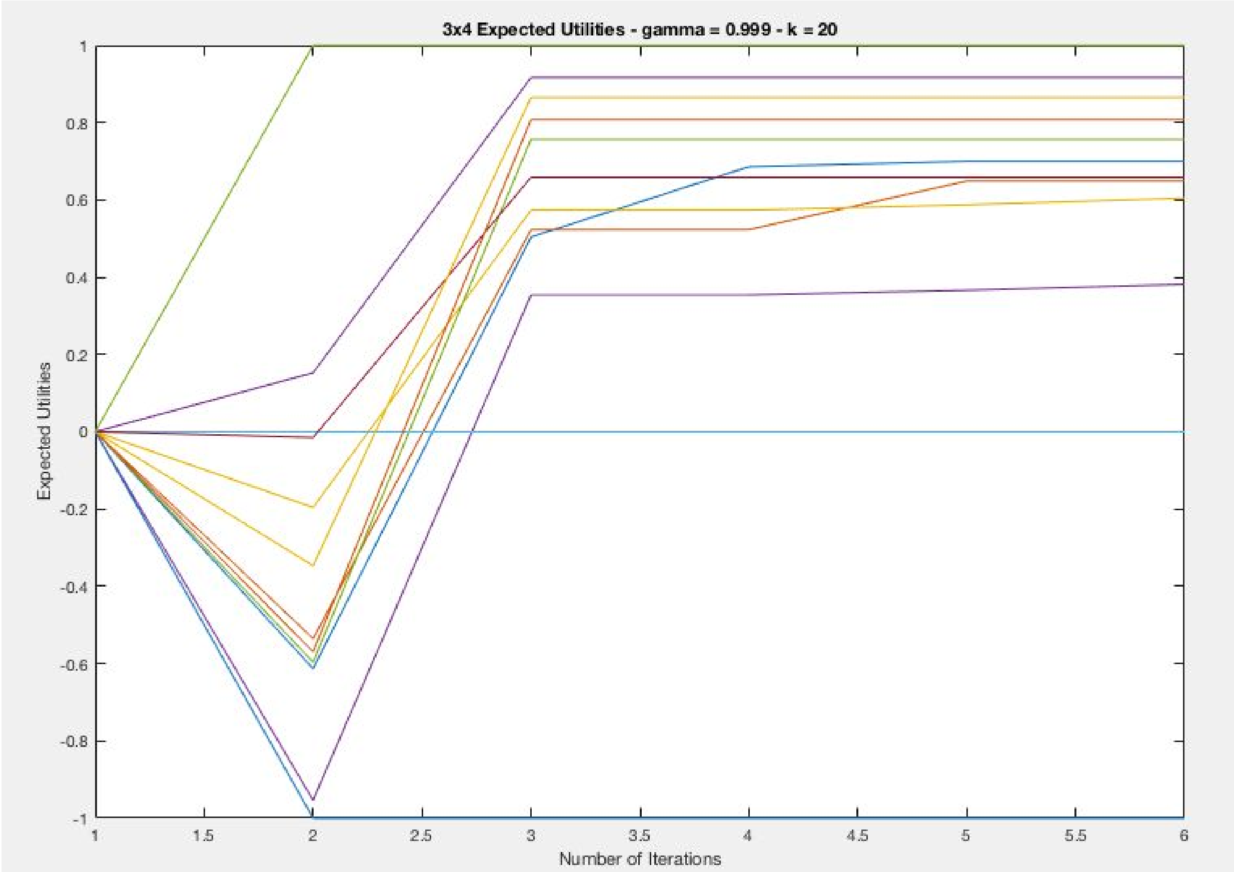
## 4. Data and Analysis

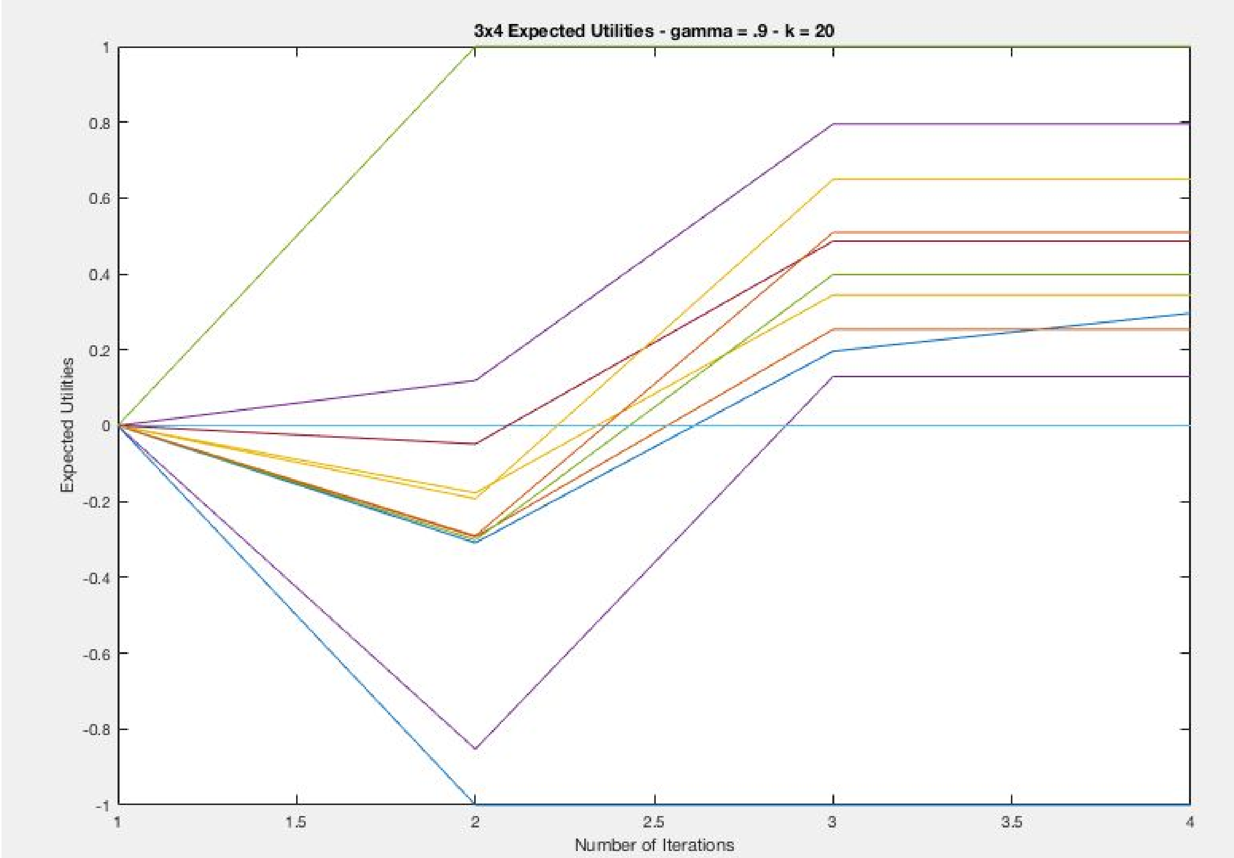



3x4 Expected Utilities Over Time - k = 10
Expected Utilities
Number of Iterations
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
1.5
2
2.5
3
3.5
4
4.5
5


3x4 Expected Utilities Over Time - k = 3
Expected Utilities
Number of Iterations
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
1.5
2
2.5
3
3.5
4


3x4 Expected Utilities - gamma = 0.999 - k = 20
Expected Utilities
Number of Iterations


3x4 Expected Utilities - gamma = .9 - k = 20
Expected Utilities
Number of Iterations

Our final MDP_Policy for the Wumpus board given in the intro:
This is for a gamma of .9 as well as a gamma for .999999
And this is for k = 3 up to k = 20

| | | | |
|---|---|---|---|
| 4 - RIGHT | 4 - RIGHT | 1 - UP | GOLD |
| 1 - UP | 2 - LEFT | PIT | 4 - RIGHT |
| 1 - UP | 2 - LEFT | WUMPUS | 4 - RIGHT |
| 1 - UP | 2 - LEFT | PIT | 4 - RIGHT |

**5. Interpretation**

As our k (max iterations) gets bigger the final policy is more precise; however, this precision quickly caps off. Five to six iterations is the stopping point for both the wumpus board and the 3x4 book board, even when given a large k such as 20. We believe this is because policy iteration is more effective at finding the best decision thanks to it's look ahead functionality.

We also tested our boards with varying degrees of gamma precision along with the changing k values. Making the gamma much more precise resulted in 2 more iterations on average for each value of k. It seems that the gamma value has a very clear effect on the utilities but little to no effect on the policy. When run for .9 and .999999 we were given the same policy. We believe that it is because even though the utilities are less precise, the ratio of their utilities is mainly unchanged for varying gamma values, leading to the same policy as a result.

**6. Critique**

Not much was different from A7 in terms of our implementation or simulation techniques. The code for policy iteration is extremely similar to value iteration and we had to change very little. The policy evaluation algorithm was also very simple to write because it followed the same structure as the iteration functions. We could have alternately tried writing the policy evaluation algorithm as a system of linear equations instead to learn a new technique instead of copying our other convention but we chose to go for simplicity and an algorithm that we fully understood.

## 7. Log

Isabelle’s Log: (Even Sections)
3 hours

Karla’s Log: (Odd Sections)
About 3 hours between the coding and lab report