248728 **Rui Carapinha**

Artificial Intelligence **Markov Decision Problems**

First Part

1.

In this part of the report I used the Value Iteration Algorithm. The parameters were: r = -0.04 and y = 1. I needed 23 iterations and the values I got after running the algorithm were the following:

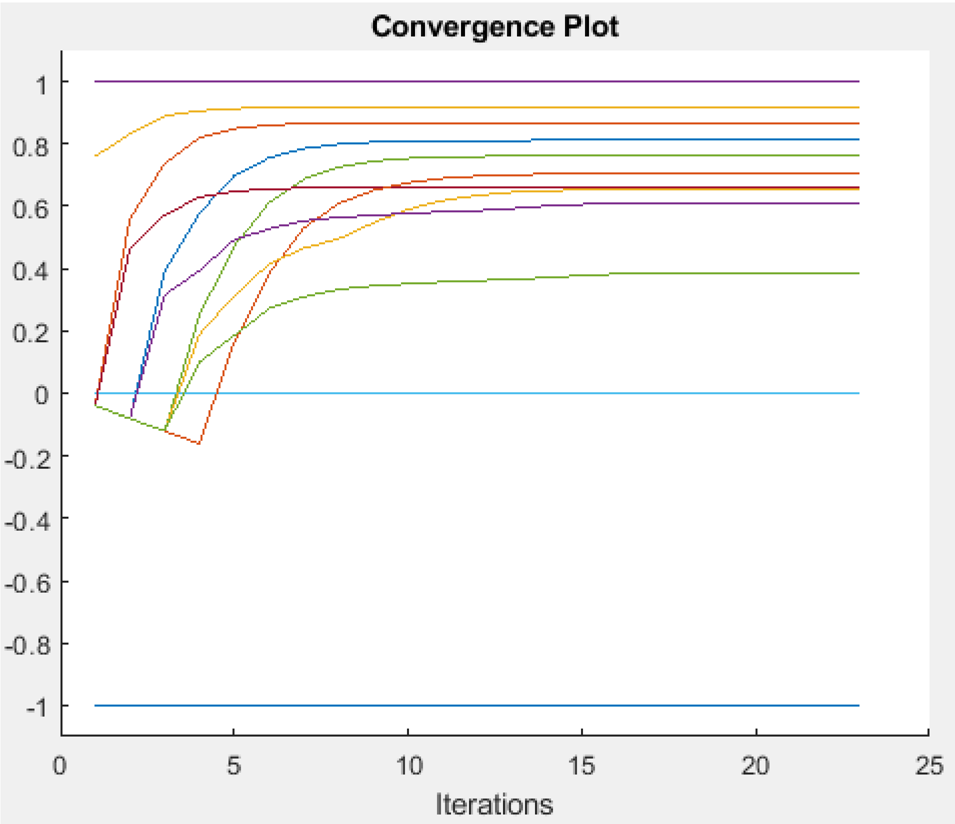
```
result =

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

Which lead to the following policy:

>	>	>	1
^	Wall	^	-1
^	<	<	<

We can see that the results correspond to the ones demonstrated in the lecture.

The convergence plot is the following:



2.

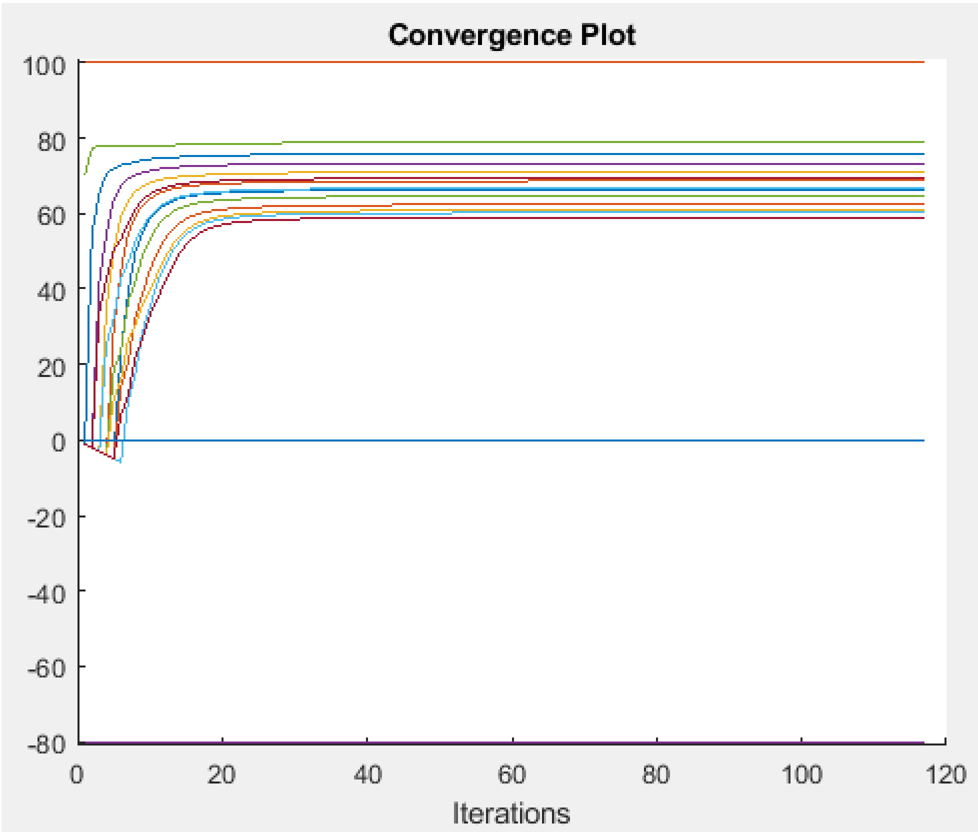
In this part of the report I used the Value Iteration Algorithm. The parameters were: r = -1 and y = 0.99. I needed 49 iterations and the values I got after running the algorithm were the following:

```
result =

   71.6296   74.0186   76.4600   78.9010
   69.8537   72.0653   74.7558   81.4650
   67.5416   65.9687  -20.0000   84.5949
   65.2988   64.0530         0  100.0000
```

Which lead to the following policy:

>	>	>	v
>	>	^	v
^	<	-20	v
^	^	Wall	100

The convergence plot is the following:



3.

In this part of the report I used the Value Iteration Algorithm with 117 iterations. In changed the special value to -80 the values I got after running the algorithm were the following:

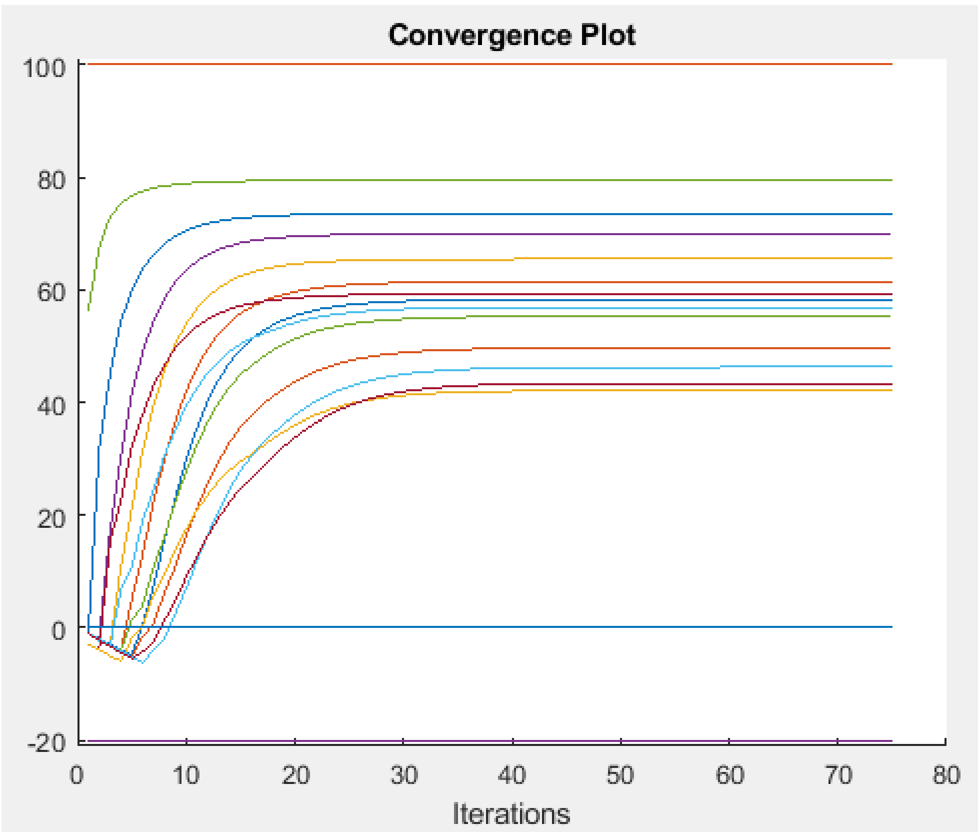
```
result =

   66.3313   68.6466   71.0126   73.3783
   64.6102   66.7536   69.3610   75.8631
   62.3695   60.8451  -80.0000   78.8964
   60.1959   58.9886         0  100.0000
```

Which lead to the following policy:

>	>	>	v
>	>	^	v
^	<	-80	>
^	^	Wall	100

The agent now will try everything to not go to the bonus position, that's the top priority now. We can see in that in the position (3,4) where the agent prefers to rely in the probability of 0.1 to go to the goal position (when we chooses to go to the right) than go straight to the goal (go down) and rely on the probability of 0.8 of that to happen (but risking the probability of 0.1 to go to the bonus position).

The convergence plot was the following:



4.

In this part of the report I used the Value Iteration Algorithm with 75 iterations. In this part of the report I changed the action uncertainty value to p1 = 0.6, p2 = 0.2, p3 = 0.1 the values I got after running the algorithm were the following:

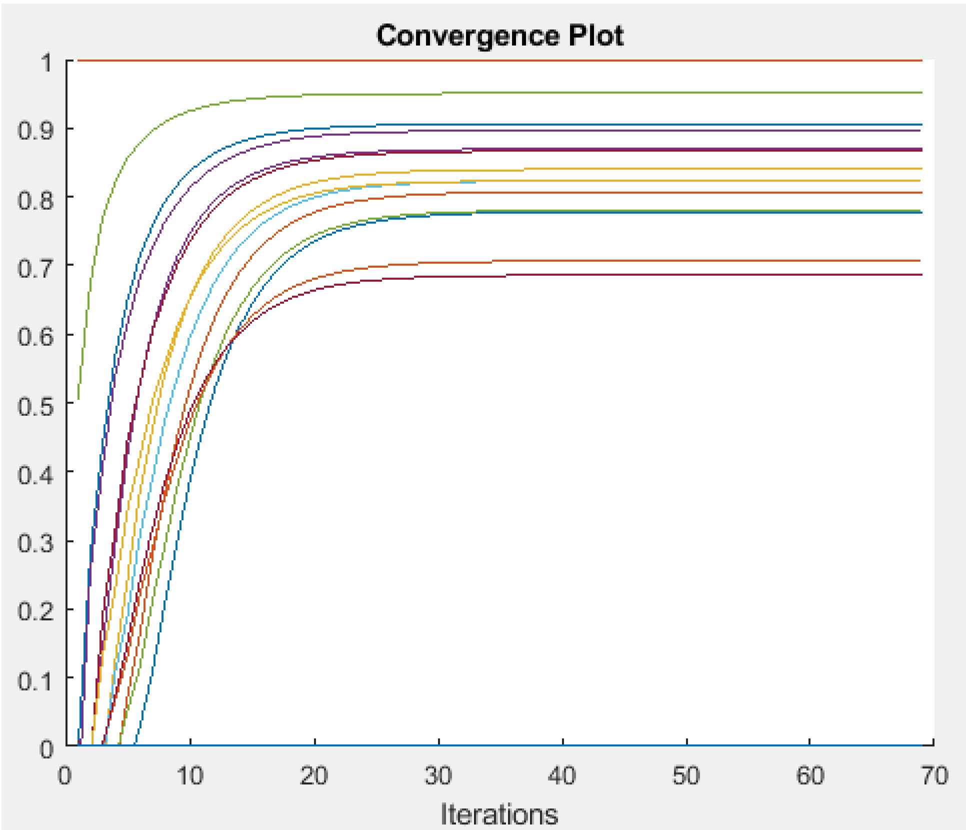
```
result =

   58.2681   61.4348   65.4420   69.9216
   55.2551   56.8645   59.2832   73.5288
   49.6402   42.0596  -20.0000   79.4256
   46.2277   43.3753         0  100.0000
```

Which lead to the following policy:

>	>	>	v
^	^	>	v
^	<	-20	v
^	<	Wall	100

There aren't a lot of changes in the policy the agent used in this part of the work. There are some different actions, but the agent will decide to go to the goal by almost the same path.

The convergence plot was the following:



5.

In this part of the report I used the Value Iteration Algorithm and I changed the discounting factor to 1.1. The program didn't stop because of the Epsilon criteria, it only stopped after all the iterations. The values I got after running the algorithm were the following:

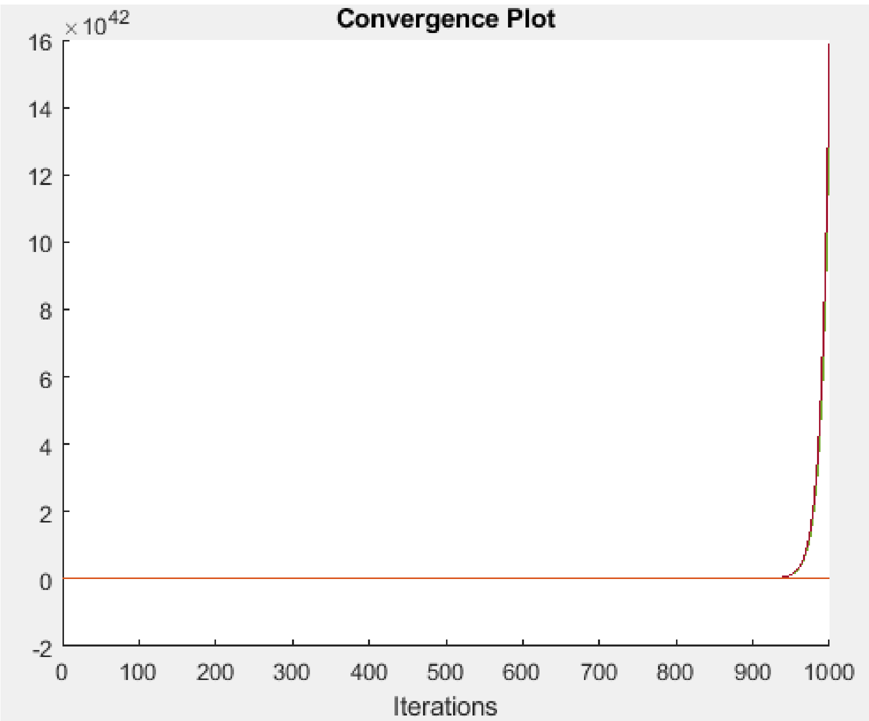
```
result =

   1.0e+43 *

    1.5908    1.5908    1.5908    1.5908
    1.5908    1.5908    1.5908    1.5908
    1.5908    1.5908   -0.0000    1.4140
    1.5908    1.5908         0    0.0000
```

Which lead to the following policy:

>	>	>	>
>	>	^	^
>	<	-20	^
>	>	Wall	100

We notice that the agent now decides that going to the Goal is bad for him, he gets more points if just goes around and making sure he doesn't go to the Goal.

The convergence plot was the following:



If I changed the discount factor to above 1 the utilities will turn to exponential and the agent would prefer to move around than to go the goal. We will get more points if he makes sure he never goes to the goal.

6.

In this program I have 2 .txt files with the Maze and Data. The file with the Maze is the following:

```
1,1,1,1
1,1,1,1
1,1,-20,1
1,1,-50,50
```

The Parameters file is the following:

```
0.8,0.1,0.1,-0.04,0.99,0.2
```

This file specifies the probabilities, the learning rate and all the other parameters.

After running the program, the Q-Learning matrix was the following:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0.5377	1.5128e+03	-1.0689	1.0933	1.5128e+03	1.4193	-0.0825	1.3546	0.7015	-0.8314	-0.2938	-0.1303	-0.3031	-0.1623	-1.5062	-1.0667
2	1.4986e+03	1.4897	1.5270e+03	1.1093	0.0859	1.5270e+03	-1.9330	-1.0722	-2.0518	-0.9792	-0.8479	0.1837	0.0230	-0.1461	-0.4446	0.9337
3	-2.2588	1.5128e+03	-2.9443	1.5414e+03	-1.4916	0.1978	1.5414e+03	0.9610	-0.3538	-1.1564	-1.1201	-0.4762	0.0513	-0.5320	-0.1559	0.3503
4	0.8622	1.4172	1.5270e+03	0.0774	-0.7423	1.5877	-1.7947	1.5560e+03	-0.8236	-0.5336	2.5260	0.8620	0.8261	1.6821	0.2761	-0.0290
5	1.4986e+03	0.6715	0.3252	-1.2141	-1.0616	1.5270e+03	0.8404	1.4367	1.4415e+03	-2.0026	1.6555	-1.3617	1.5270	-0.8757	-0.2612	0.1825
6	-1.3077	1.5128e+03	-0.7549	-1.1135	1.5128e+03	0.6966	1.5414e+03	-1.9609	0.5080	1.4747e+03	0.3075	0.4550	0.4669	-0.4838	0.4434	-1.5651
7	-0.4336	0.7172	1.5270e+03	-0.0068	-0.6156	1.5270e+03	0.1001	1.5560e+03	0.2820	0.5201	-1.2571	-0.8487	-0.2097	-0.7120	0.3919	-0.0845
8	0.3426	1.6302	-1.7115	1.5414e+03	0.7481	-0.2437	1.5414e+03	-1.2078	0.0335	-0.0200	-0.8655	1.5900e+03	0.6252	-1.1742	-1.2507	1.6039
9	3.5784	0.4889	-0.1022	-0.7697	1.4350e+03	0.2157	0.3035	2.9080	-1.3337	1.4747e+03	-0.1765	0.5528	1.4281e+03	-0.1922	-0.9480	0.0983
10	2.7694	1.0347	-0.2414	0.3714	0.8886	1.5270e+03	-0.6003	0.8252	1.4415e+03	-0.7982	0.7914	1.0391	-1.0298	1.4415e+03	-0.7411	0.0414
11	-1.3499	0.7269	0.3192	-0.2256	-0.7648	-1.1480	0.4900	1.3790	0.3502	1.0187	-1.3320	-1.1176	0.9492	1.5301	-0.5078	-0.7342
12	3.0349	-0.3034	0.3129	1.1174	-1.4023	0.1049	0.7394	1.4548e+03	-0.2991	-0.1332	-2.3299	1.2607	0.3071	-0.2490	-0.3206	1.6241e+03
13	0.7254	0.2939	-0.8649	-1.0891	-1.4224	0.7223	1.7119	-0.4686	1.4415e+03	-0.7145	-1.4491	0.6601	0.1352	1.4415e+03	0.0125	0.2323
14	-0.0631	-0.7873	-0.0301	0.0326	0.4882	2.5855	-0.1941	-0.2725	-0.2620	1.4747e+03	0.3335	-0.0679	1.4281e+03	1.6035	-3.0292	0.4264
15	0.7147	0.8884	-0.1649	0.5525	-0.1774	-0.6669	-2.1384	1.0984	-1.7502	-0.2248	0.3914	-0.1952	0.2614	1.2347	-0.4570	-0.3728
16	-0.2050	-1.1471	0.6277	1.1006	-0.1961	0.1873	-0.8396	-0.2779	-0.2857	-0.5890	0.4517	1.6088e+03	-0.9415	-0.2296	1.2424	-0.2365

This matrix represents the Q-Values using the Bellman's equation. In the rows we have the current state and in the columns, we have the next state. Of course, some next states are unreachable depending on the current state, those are the one which have really low values.

The Q-Learning policy I got for the map was the following:

0	0	0	0
1	1	1	1
1	0	0	1
1	0	0	Goal