Maximilian Bragazzi Ihrén maxbr431@student.liu.se

TDDC17 – Lab 4

Part II

7) The **states** are simply the discretization of the angle in the input.

“`P_STATE: 2`” is an example of a state. Since, after the discretization, we want a fairly small number of states left (which would be the nrValues value), I started with a nrValue of about 10. While functional, I figured I should try going smaller. I ended up at 7, and stopped there (mostly due to time constraints). And since we want the rocket facing upwards, and the angle seems to go between 0 (upwards) and PI (downwards), I figured -PI/2 (facing left) and PI/2 (facing right) would be good values for min and max.

The **reward** is simply the negative value of the absolute of the current angle. This is due to a want for keeping the reward function simple, and to always have the agent get better. Since the angle 0 is upwards, it will strive towards getting lesser negatives, which in turn will leave it pointing upwards eventually.

The **Q-Update** function works in such a way that sets the current q-value to the old q-value added to the learning rate, multiplied by an expression which adds the current reward to the best reward (multiplied by the discount factor), and subtracts the old q-value.

Since the learning rate uses the alpha function provided, it decreases the learning rate over time (at the rate given in the example code, 10).

8) If I turn off exploration over all (by replacing the “exploration” variable in the selectAction function with false), it still tends to find a solution in a slightly smaller time frame (which might just be a coincidence from the few iterations I tried it on).

The point of the exploration is to try to find a more optimal Q-function (considering more/different states than it would otherwise). When disabling the exploration, you end up in a situation where the agent will stop learning after X iterations. This might be good (the agent ends up facing up), or bad (the agent faces down and can't get to an upwards state).

Maximilian Bragazzi Ihrén maxbr431@student.liu.se

Part III

For the hover part I didn't change much in the existing code (besides the function calls, of course). I changed the calculation for the stateAngle (not in the getStateAngle function, but I copied the code and made the changes), so that it uses PI/4 instead of PI/2 for its min/max values.

In getStateHover I use the discretize2 function for all three values, and put them all together in a string (separated by their respective variable names).

In getRewardHover I get the rewards for the velocities the same way I get the angle. And I felt that the angle being correct had more impact on the actual correctness of the simulation, so I added some weight to the angle reward before merging them together.