# Report for Lab 5 in TDDC17, Artificial Intelligence
## *Reinforcement Learning*

Margareta Vi
marvi154

Markus Petersson
marpe163

October 11, 2016

# 1 A comment on the actions available to the rocket

In both task 2 and 3 the rockets have the same set of actions available. The available actions are all combinations of turned off and on engines, i.e. 8 different states.

# 2 Q-Learning Angle Controller

In this section we will answer the given questions in the lab regarding the Angle controller.

## 2.1 Motivation of choices of states and reward function

In task 2 the number of states the rocket can be in is 16, based on cardinal directions and its reward function can be seen in Equation 1.

$$Reward = 1 - \left|\frac{angle}{\pi}\right| \tag{1}$$

The choice of reward function is such that the bigger the angle is the less reward the rocket will get. This to make it strive to have angle 0.

## 2.2 Purpose of components of the implemented Q-learning update

In Equation 2 we see how the Q-values are updated in each state, depending on a certain action and the reward of being in that state.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)) \tag{2}$$

$Q(s,a)$ is the Q-value in the previous state, $s$, and doing the previous action $a$. $R(s)$ is the reward function, which depends on which state we are in. $Q(s',a')$ is the Q-value in the new state $s'$ and current action $a'$. $\alpha$ is the learning rate, i.e how fast the rocket learns and $\gamma$ is the discount factor which tell us if we should focus on short term rewards or long term rewards. $Q(s,a)$ is a element in the hashtable, which says how well the robot performed in a certain state given a certain action (the reward is included in the Q-value).

## 2.3 Result of neglecting training

Since the rocket has had no opportunities to learn from experience the behaviour when turning off exploration before learning is random. In the lab, it is hard to not let the rocket train at all, so if it gets to train for one or two states it will depend on those few training times though it might still behave randomly.

# 3 The Full Q-Learning Hover Controller

In this section we will answer the given questions in the lab regarding the Hover controller.

## 3.1 Motivation of choices of states and reward function

For the states used in task 3 we chose to utilize the given *discretize* method, in order to obtain a finite set of states. All such states were then associated with a unique string in order to label the state.

The reward function for task 3 was designed to introduce rather large steps in the reward to distinguish between desired states and undesired ones. In particular, we reward desired angles and velocities with quite a bit of rewards, while bad velocities causes a large drop in reward, that grows with how large the velocities are. We present this reward function using the code used in the lab. The code can be found in Listing 1.

Listing 1: Code for the reward function in task 3

```java
public static double getRewardHover(double angle, double vx, double vy) {
        double reward = 0;

        reward = reward - 2*Math.abs(angle/Math.PI);
        reward = reward + 2/(1+Math.abs(vx));
        reward = reward + 2/(1+Math.abs(vy));

        if (Math.abs(angle/Math.PI)<0.05){
            reward = reward + 10;
        }
        if (Math.abs(angle/Math.PI)<0.005){
            reward = reward + 25;
        }
        if (Math.sqrt(vx*vx+vy*vy)<0.05){
            reward = reward+10;
        }
        if (Math.abs(vy)>0.2){
            reward = reward-50*Math.abs(vy);
        }
        if (Math.abs(vx)>0.2){
            reward = reward-50*Math.abs(vx);
        }
        if (Math.sqrt(vx*vx+vy*vy)<0.005){
            reward = reward+25;
        }

        return reward;
    }
```

## 3.2 Purpose of components of the implemented Q-learning update

The Q-update equation for task 3 is still Equation 2 and those variables means the same. The difference is that the reward function for task2 and task3 are different.