USING DEEP Q-LEARNING TO OPTIMIZE ROCKET LAUNCH VEHICLE PERFORMANCE

Shiming He
Bluevale Collegiate Institute
MehtA+

Nayana Jayakumar
Hillsborough High School
MehtA+

Ishan Venkatesh
Monte Vista High School
MehtA+

Jason Huang
Columbian High School
MehtA+

August 24, 2024

ABSTRACT

Spaceflight is one of the greatest engineering marvels humanity has developed. The first space missions began with mathematicians, scientists, engineers, and countless others who worked tirelessly to send the first vehicles into space. Since then, new opportunities has appeared in which modern technology improves the performance of launch vehicles (rocket) and spacecraft. This project continues this legacy by researching the use of reinforcement learning in launch vehicle trajectory optimization. Rocket trajectory optimization is an interplay of many different factors and is difficult to optimize to produce the best performance. To solve this problem, this project aims to use reinforcement learning model, Deep Q-learning (DQL), to create a model that allows for the optimization of launch trajectory where the goal is lifting a set payload to the highest and most circular orbit. The model in this project demonstrated the capability of DQL in optimizing the launch trajectory by providing a path to orbit that allows for better orbital performance under certain circumstances.

1 Introduction

The launch trajectory of a rocket is vital in the performance of launch vehicle. In an industry where every kilogram matters, achieving an optimal launch trajectory would allow for longer longevity of the payload, where the payload can be inserted into a deployment orbit where less fuel will be used by the payload in raising to its target orbit.

Though there is a good understanding of the creation of the launch trajectory of a rocket, optimizing this trajectory is an intricate problem, requiring high-level math and the support of computers. Launch trajectories have improved since the Space Race in the 1960s, with more complex trajectories being used for different launch vehicles. However, this problem of improving its launch trajectory's efficiency persists to this day.

Therefore we propose the use of Deep Q-learning as a solution to optimize the launch trajectory of launch vehicles. Deep Q-learning also has the potential, once a policy is created, of real-time optimization of a trajectory mid-flight using low computational power. This would be beneficial in situations, like the launch failure of Starlink Group 9-3 on June 11, 2024. In this case, a DQL policy would have allowed the trajectory of the second stage of the Falcon 9 launch vehicle to change once the anomaly occurred to allow it to reach an intermediate orbit, which would have allowed a more successful deployment of the Starlink Satellite without the rapid orbital decay that were observed.

2 Related Work

The work related to the use of Reinforcement Learning (RL) in aerospace for trajectory optimization is sparse. It was used in a 2020 thesis *Autonomous Guidance For Multi-Body Orbit Transfers Using Reinforcement Learning* by *Nicholas B. LaFarge* from Purdue University, which describes the use of RL to create a flight control system for small satellites following interplanetary trajectories while using low computational resources[1]. Compared to our project, this system does not optimize the performance of a trajectory, the key focus of our project; rather, it focuses instead on following a preset trajectory for small-sized satellites beyond low earth orbit missions, which face a different set of challenges compared to a launch of a rocket from earth. In another project, RL model, Deep Q-learning, is used

in the optimization of landing a lunar lander in a preset environment by OpenAI. Compared to our own research, the Lunar Lander DQL[2] does not face the dynamic and complex environment in rocket launches and can therefore have simpler reward functions, which would not be the case for this project. Our research is based on the combination of the strengths of the two studies, including similar states to the work of *Nicholas B. LaFarge*, and a similar DQL model from the Lunar Lander Project.

3 Methodology

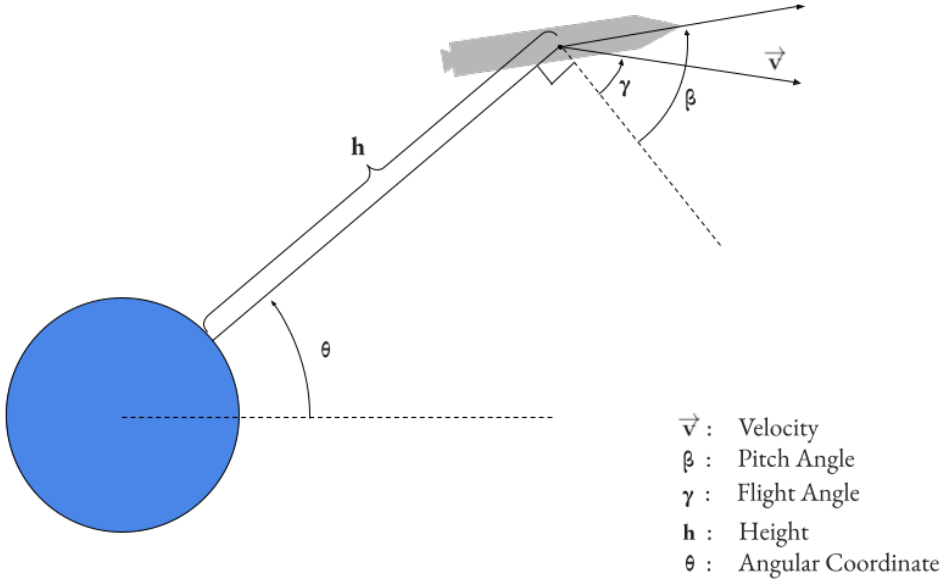

Figure 1: Launch Variable Definition

3.1 Environment

The environment for this project was constructed from scratch. Using the properties of a given rocket, the trajectory is constructed using a changeable pitch angle that can be modified externally. This environment relies on Newtonian physics to generate the next state of the model with the capability of finding the perigee[1], apogee[2], and eccentricity[3] of the present orbit. The environment is set to update/step every 0.1 seconds to allow balance between the accuracy and the speed of the simulation. The custom environment allowed our group to generate states that were found to be most applicable to our project, which changed based on the progress that was made.

Please refer to the Appendix for additional details.

3.2 Model

Based on research by our group, it was determined that Deep Q-learning (DQL) would be the best model to use for this project. Deep Q-learning was designed to work in continuous state and discontinuous actions, which is the condition of our environment making it a good fit for this project[3]. Moreover, as mentioned in related works, similar projects like the DQL Lunar Lander also used DQL as the Reinforcement Learning technique, allowing us to have a example to refer to. The combination of these two factors made DQL appealing to our group.

[1]Perigee: the closest point to earth of an orbit.
[2]Apogee: the farthest point to earth of an orbit.
[3]Eccentricity: defines the circularity of an orbit. 0 is a circle and 1 is a parabola.

3.3 Actions and States

Based on the environment, the actions determined for controlling the trajectory of the launch vehicle for the DQL model was the change in the launch vehicle's pitch angle. It was decided that the actions were going to be pitch up, stay at the same pitch, and pitch down. Due to the preset 3° per second pitch rate, the resulting actions mentioned above would change the pitch angle by +0.3° (up), 0.0°, or -0.3° (down) in each update interval.

The state of the model was originally a 10-dimensional state vector, however following many experiments on the size of the state vector it was determined that the state vector had to be decreased for the Q-loss to converge given our limited computational capability. According to the states used in the thesis by *Nicholas B. LaFarge*, the states were decreased to velocity and position in both axes in the Cartesian plane; thrust; and mass[1]. This would later follow additional changes that modify the plane of observation to polar coordinate plane, as well as encompassing the present orbital eccentricity and time. The polar coordinate plane was chosen above that of the Cartesian plane, due to the velocity in the direction that is parallel to the surface of the earth, is key in determining the orbit of the payload.

Please refer to the Appendix for additional details.

3.4 Q-function

This project uses a Deep Q-Network (DQN) for the Q-function of the DQL model. This neural network was originally sized at 3 layers, however following much experimentation, it was found that the environment and the reward function made the Q-function complex and dynamic resulting in the need for extra neural network layers to compensate. This research finalized on a neural network with 21 layers, which was determined to be fitting for our project.

3.5 Learning Rate and Loss Function

Based on the experimentation between different learning rates, a low learning rate of 10^{-7} was chosen. This is because higher learning rates caused to great of a step sizes, which led to the model not being able to converge. MSE Loss is used for this research. Smooth L1 Loss was also tested, however it was not able to offer the performance of MSE Loss.

Please refer to the Appendix for additional details.

3.6 Model Optimization Period

Initially, the goal was to optimize the entire trajectory of a launch vehicle's launch. However, due to dynamic situations of the launch, it was decided that this research would instead use the last 10 and 20 seconds of the launch vehicle's launch to run the DQL model to optimize the performance from a standard trajectory. The previous portions of the trajectory would be instead simulated based on the standard trajectory.

Please refer to the Appendix for additional details.

3.7 Reward Function

Unlike other situations, the rocket launch trajectory does not offer the ability to define explicit intermediate rewards. For example, there is no reward that will clearly guide the rocket to achieve an optimal trajectory. Instead, we decided to define vague intermediate rewards in an attempting to lead the model to reach the optimal orbital performance.

The following example is the reward function for the 10-Second Optimization model.

$$R_{intermediate} = \frac{0.01}{(1+|\gamma|)^2} - \begin{cases} 0.003(N-3) & N > 3 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$R_{final} = \frac{3.125 \cdot 10^2}{(1+e)^8} + 25 \cdot \frac{p - 227000}{3000} - \begin{cases} 0.03 & N = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

γ is in degrees.

See Figure 1 for variable explanation.

The intermediate reward contains two parts. The first part is a piece-wise function which will allow for a maximum of 0.01 reward per step (every 0.1 seconds), which will decrease as the flight angle (γ) becomes farther from 0°. This allows the launch vehicle to attain a γ of 0°, where it gains the most velocity, which will largely determine the final

orbit. The second part is also a piece-wise function that gives a negative reward based on the amount of change in pitch change direction (N) [4] up to the moment in time for which this reward was calculated, after more than $3N$ has occurred. This is done to decrease the amount of oscillation the model would create in its resulting trajectory.

The final reward contains three parts. The first part gives a reward based on final orbit eccentricity. This allows the model to target lower eccentricity (e) (more circular) orbits. The second part gives a reward based on the height of the perigee (p). Due to the short optimization period for the 10-Second Optimization model, the function has a large slope to induce a higher perigee orbit. The last part of the final reward gives a low negative value for a trajectory that would experience no change in pitch angle to ensure that the model would not get stuck at a local maximum at a constant 0° pitch angle.

Please refer to the Appendix for additional details of the 20-Second Optimization reward function.

4 Results and Discussion

4.1 General Results

Our model was able to optimize the performance of the final orbit as per Table 1. In both cases, the model resulted in a simplified trajectory, where the majority of actions were the same. In other cases where the model converged onto a different value, similar simple trajectories were observed. In all cases, the change in pitch change direction was below the limited amount specified in the intermediate reward section. Therefore it is unlikely that the intermediate reward is causing this simplification of the trajectory.

Table 1: Orbital Performance

	Perigee (km)	**Apogee (km)**	**Eccentricity**
Original Trajectory	227.325	3859.163	0.21581
10-Second Optimization	232.554	3637.829	0.20498
20-Second Optimization	227.793	3857.892	0.21572

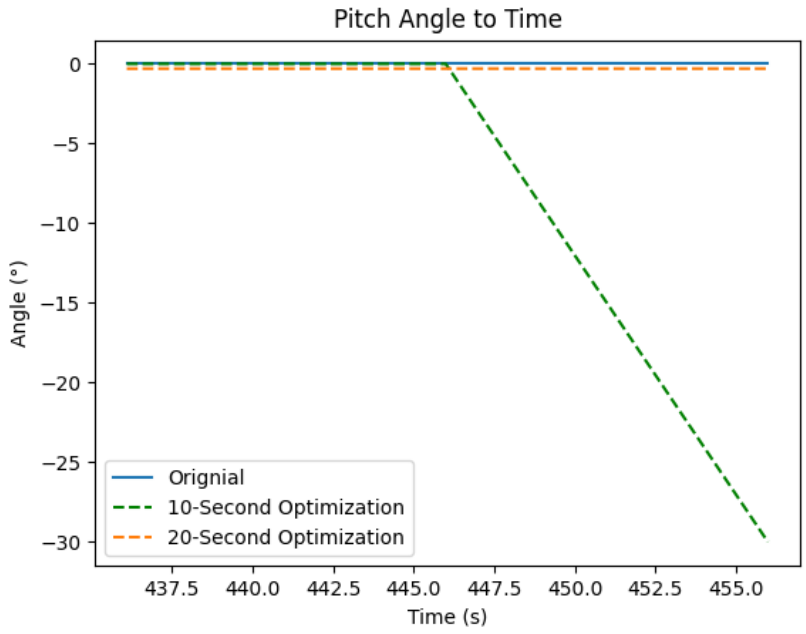

Figure 2: Pitch Angle to Time For Corresponding Results

[4]The change in pitch change direction counts only when the launch vehicle pitch changes from holding the same pitch to pitching up or down; or when direction at which the pitch changes flips.

4.2 10-Second Optimization Result

The best performance was observed in the 10-Second Optimization Results. The model was able to determine that to achieve the best trajectory it had to decrease the pitch angle constantly for the 10 seconds in which the model was trained (see Figure 2). Though successful in improving final orbital performance, it was also observed that during training higher reward outcomes were achieved; however, they were not capitalized upon. Instead this simple trajectory with only pitch down action was the result of the model, when it reached convergence. The resulting trajectory of our model resulted in an increase of perigee by 5 km and eccentricity reduction by 5%. This is a good result, with perigee and eccentricity nearly reaching its optimization limit given the circumstance of this test case.

4.3 20-Second Optimization Result

The results observed in trajectory optimization in the 20-Second Optimization Results were not as successful as those observed in the 10-Second Optimization. Similar to the results noticed in the 10-Second Optimization, the model decided upon a simple trajectory, holding a -0.3° pitch angle throughout the flight (see Figure 2). Though this does produce a difference in final orbital performance, with the perigee being increased by 470 m and the eccentricity decreased by 0.04%, it is a marginal improvement and is far from the performance reached by the 10-Second Optimization version. It was expected that the increase in performance observed in the 20-Second Optimization was to be similar or higher than that of the 10-Second Optimization, due to its capability of replicating the total trajectory of the 10-Second Optimization version. However, the results showed the opposite.

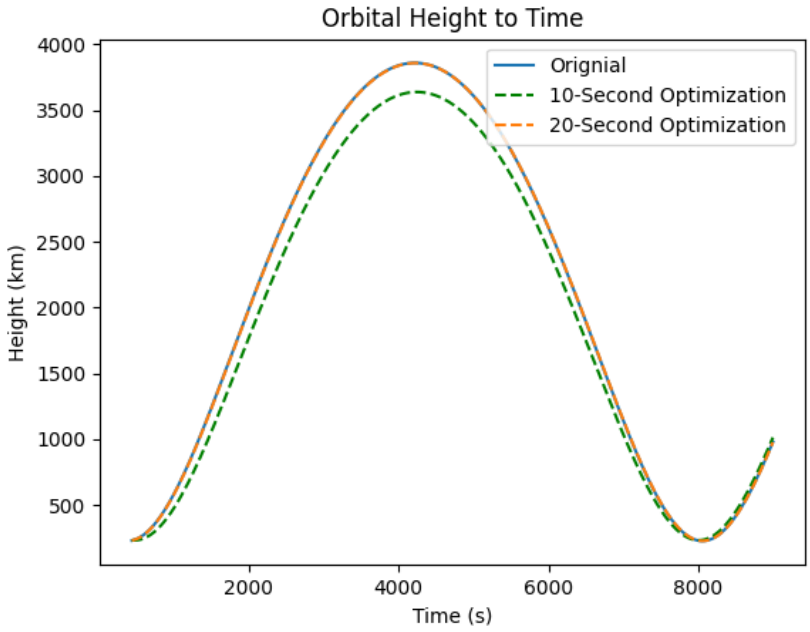

Figure 3: Orbital Height to Time For Corresponding Results

4.4 Discussion

In both cases, the DQL model was able to converge onto a single trajectory for a launch trajectory after between 400 and 800 episodes of training. However it is important to note that for the case of 10-Second Optimization, it was observed that the model at certain times converged on other rewards that were lower than the maximum reward. It is also important to note that for both cases, as well as other experiments with the same model and reward function, our model results converged to a simple trajectory, which consisted of up to 2 changes in pitch change direction, with the majority having one or no changes in pitch change direction. This is interesting since this convergence made the model not able to converge on the highest possible reward trajectory, even when higher reward trajectories were reached. On top of that, in cases where the most simple trajectory would be made disfavorable through extra negative reward, the model would still try to converge onto another simple trajectory or achieve something extremely similar that bypasses

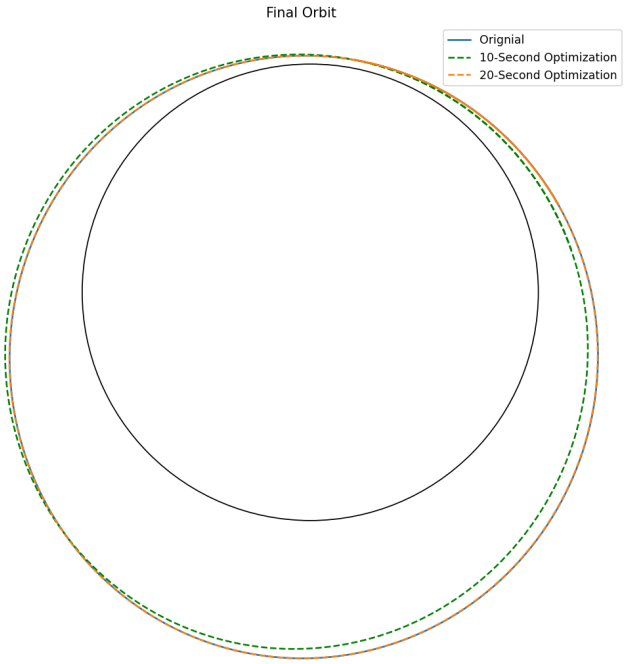

Figure 4: Final Orbit For Corresponding Results

the negative reward. This persistence in reaching such simple trajetories is the largest hurdle that this model faces for further optimizing its performance. This is most noticeable in the case of the 20-Second Optimization. Since in that example the trajectory from the 10-Second Optimization, with no action for the fist 10 seconds, may be used in the 20-Second Optimization case, which would result in a 70% increase in reward compared with the present result. Further it is likely that the 10-Second Optimization trajectory is not the most optimal trajectory for 20-Second Optimization either. We conjecture that this could be caused by certain characteristics of the reward function, neural network, or the training method, though it is most likely either the intermediate reward somehow providing more short term positive reward for such a simple trajectories [5] or too simple of a neural network. Solving this problem would be a major focus for future development of this project.

5 Conclusion

Overall, the results of our model demonstrates the concept of using Deep Q-learning in optimizing a rocket launch vehicle's performance. Using only a short time period during the launch, the DQL Model was able to improve the performance of the final orbit. These results show promise for future development of Deep Q-learning in optimizing launch trajectory.

6 Future Work

In the future, continual improvement of the existing model in trying to overcome the challenges faced by the present model would be key. However another key point would be the experimentation of new reinforcement learning model in place of Deep Q-learning. Originally, DQL seemed to be the most optimal model for our project due to its continuous states and discontinuous action design. However following further development of the project, we encounter multiple challenges in the use of Deep Q-learning, especially with the failure to generate more complex and optimal trajectories. Due to this, experimentation of new models, like the Action-Critic model which uses continuous states, would be a central part in future work[4]. Additionally, future development wil also be focused on creating more appropriate reward function, increasing computational resources, and experimentation of continuous states with new models; in an attempt to increase the model's performance.

[5]This is at the same time unlikely, since the only other section on top of the one decreasing the change in pitch change direction, is a reward for lower flight angle, which a simple trajectory would not be able to maximize in.

7 Division of Work

1) Building the simulation - Shiming H.

2) DQL Model - Shiming H., Nayana J.

3) Reward Function - Shiming H., Ishan V.

4) Optimizing the model - Shiming H., Nayana J., Ishan V.

5) Creating the poster - Shiming H., Nayana J., Ishan V. , Jason H.

6) Writing the paper - Shiming H., Nayana J., Ishan V. , Jason H.

8 Acknowledgements

This team would like to thank our instructors, Miss Haripriya Mehta, Mr. Bhagirath Mehta, and Mr. Shokhruz Kakharov. We appreciate their guidance. We were fortunate to have had the opportunity to learn from them, and thanks to the lessons we were taught, we were able to create this project. Thanks to them, this project has grown.

References

[1] Nicholas B LaFarge. Autonomous guidance for multi-body orbit transfers using reinforcement learning. Master's thesis, Purdue University, 2020.

[2] Alex. Dqn on lunar lander - rl, Jul 2023.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[4] Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee, 2020.

[5] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

9 Appendix

9.1 Environment

This environment produces a trajectory based on the written performance of a launch vehicle passed in as a JSON file, as well as the change in pitch angle, which is passed in on each step. The environment models multiple different factors of a rocket's performance, including the launch inclination, the drag force of the rocket when in the atmosphere, the change in mass, change in gravitational acceleration, and gravity turn (this is not used for this project due to the complexity it would bring). The launch inclination is passed in when the environment is initialized along with the JSON file for the rocket performance. The drag force (D) acting on the rocket uses the barometric formula to find the atmospheric pressure alongside an estimated drag coefficient equation to find the force experienced by the launch vehicle. The change in mass (M) is determined based on the timestamp, burn rate, and stage separation time to find the corresponding mass. The thrust of the rocket (T) is obtained through the use of the present burn rate and the present burn ISP (Specific Impulse). The gravitational acceleration (g) is obtained using Newton's Universal Law of Gravitation. The acceleration ($\overrightarrow{a}$), velocity ($\overrightarrow{v}$), and position ($\overrightarrow{P}$) of the rocket in the Cartesian field is obtained by the following formulas:

$$\overrightarrow{a} = \begin{bmatrix} (\cos(\beta) \cdot \frac{T}{M}) - (\cos(\gamma) \cdot \frac{D}{M}) + (\cos(\theta) \cdot g) \\ (\sin(\beta) \cdot \frac{T}{M}) - (\sin(\gamma) \cdot \frac{D}{M}) + (\sin(\theta) \cdot g) \end{bmatrix} \tag{3}$$

$$\overrightarrow{v} = \overrightarrow{v} + \Delta t(\overrightarrow{a}) \tag{4}$$

$$\overrightarrow{P} = \overrightarrow{P} + \Delta t(\overrightarrow{v}) \tag{5}$$

β, and γ are in reference to Cartesian plane.

See Figure 1 for variable explanation.

These values are then converted to polar coordinate system values, before being returned as states for the DQL model.

9.2 States

Originally, the states consisted of a 10-dimension state vector, which encompassed many extra factors not including those in the present version. However, this version increased the time required to complete training and did not succeed in making the model converge. Therefore it was decided to decrease the state vector to a dimension of 6 using thrust, mass, and velocity and position in the Cartesian plane. This was based on the states presented in the thesis by *Nicholas B. LaFarge*[1]. However, following a low convergence percentage with this state vector, the state was reorganized to contain new values which were critical in both in the simulation and in the reward function, which would eventually lead to the final 7-dimension state vector. The seven states that are used are time, thrust, mass, height, velocity in both axes of the polar coordinate system, and present orbital eccentricity. Time was originally not part of the state, since it was thought that it would bias the system in relying on time stamps. However, it was later determined that this was not a major concern and instead time would help in allowing the model to reach convergence. The states of thrust and mass were chosen since both are factors that change during the launch, therefore it would be important to have them as part of the states. As explained in Methods, velocity in the polar coordinate system was chosen for its important connections with the final orbital performance. The height was chosen as it is important in identifying the energy and position of the launch vehicle. The orbital eccentricity allows the model to know the present state of the launch vehicle's orbit.

9.3 20-Second Optimization Reward Function

Due to the extended length of the 20-Second Optimization model optimization time, the variation of the end-resulting orbital parameters caused there to be a need to rewrite the reward functions.

$$R_{intermediate} = \frac{0.01}{(1+|\gamma|)^2} - \begin{cases} 0.003(N-6) & N > 6 \\ 0.001 & A = A' \neq 0 \end{cases} \tag{6}$$

$$R_{final} = \frac{3.125 \cdot 10^2}{(1+e)^8} + 25 \cdot \frac{p - 227000}{8000} - \begin{cases} 600 & p < 100000 \\ 0 & \text{otherwise} \end{cases} - \begin{cases} 0.03 & N = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

γ is in degrees.

See Figure 1 for variable explanation.

The new condition in the piece-wise function for intermediate reward was added to give positive reward for more varying actions, in order to allow the model to reach a more optimal trajectory. This was done in conjunction with increasing the amount of change in pitch change direction to 6 before a negative reward was administered to help allow for a varying trajectory. However, both of these changes proved to be unsuccessful.

The final reward function addition is to give a large negative reward for final orbits that will reenter Earth's atmosphere (an unsuccessful orbit). This is added, because compared with the 10-Second Optimization version, the 20-Second Optimization has a possibility of the trajectory reentering earth atmosphere (an unsuccessful orbit). Therefore, this negative reward is added to dissuade the DQL from choosing such a path.

9.4 Ineffective Approaches

During the main part of the development of this project, the DQL model faced great problems with the convergence of the results. To solve this problem many methods were tested, however many proved ineffective.

9.4.1 Large Neural Networks (DQN)

The initial version of the neural network in the DQN neural network consisted of 3 layers, however, this model made it difficult for the model to converge. Small increases in the number of layers were first experimented with 5 layers and 10 layers, however this only proved slightly effective. But following the increase of the neural network to 15 layers, a significant increase in convergence was observed. However, from this point, further increases of neural networks up to 90 layers failed to provide any substantial performance improvement, and instead slowed the training rate for each episode down dramatically. The final 21 layer neural network was chosen as a balance of better performance and faster training rate, allowing it to be more effective. In future smaller layer models should be tested to see if the neural network is able to function with less layers, while still having the complexity that is required.

9.4.2 Double DQN

The Double DQN was tested in place of the DQN in the DQL model for the 20-Second Optimization version of the model, to test if this change would allow the model to converge on a more optimal trajectory[2, 5]. Yet, no noticeable improvement in performance was noted.

9.4.3 Learning Rates

Early in the project, higher learning rates were tested to see if they had an impact in allowing the model to converge. However, no noticeable effect was found. It was only later that we observed a lower learning rate helped the model achieve convergence, which resulted in the results mentioned in this research. This is likely due to the step size required to allow the model to reach convergence being a small value, therefore requiring a lower learning rate. The downside of the lower learning rate is that the model is easier to get stuck at a local maximums for rewards. In the future development of this project, work on finding a suitable learning rate should be done, to allow for quicker and more optimal trajectories.

9.4.4 Number of Training Episodes

During development of the DQL model, different amounts of training episodes were used to improve its capability. It was initially thought that the more training episodes the more likely the model would reach convergence. However, this proved to be false. It was noted that there was only a slight improvement in performance after large episode numbers, if the model itself is already converging in lower episodes. Otherwise even after large amounts of training episodes, convergence would not be reached. Based on this project's DQL model, convergence of a model would usually be able to be observed by around 400 episodes.

9.4.5 Model Optimization Time Period

This DQL model was originally envisaged to optimize the entire trajectory of a rocket launch, however, this was later found to be difficult to achieve with the resources available. Due to this, the model optimization time period was reduced to the last 10 seconds and 20 seconds of launch to test the concept of using DQL in improving launch vehicle performance. This turned out to be successful. But even in this example it can be seen that the shorter 10-Second Optimization model was able to provide better results. Therefore, it can be seen that a shorter run time would likely lead to better results with the same amount of training. In the future, a longer model optimization time period that will still achieve convergence would be a important development point to allow for the use of the true potential of Deep Q-learning in performance optimization for launch vehicles.