# Launch Vehicle Attitude Controller via Reinforcement Learning

Justin Tam[1]

[1]*Department of Electrical Engineering, University of Ottawa, Ottawa, ON, Canada*

[1]jtam050@uottawa.ca

*Abstract*— **Deep reinforcement learning as a systems controller has been growing in popularity in recent years due to its ability to handle complex environments and unexpected disturbances. The purpose of this paper will be to investigate recent developments in deep reinforcement learning of control systems and replicate one of these recent developments. As a result, a one-dimensional launch vehicle attitude controller is designed and simulated via reinforcement learning and the TD3 algorithm. The simulated results show adequate performance in controlling the attitude in the presence of randomized initial starting conditions.**

***Keywords***— **Reinforcement Learning, Attitude Control, Control Systems, Twin Deterministic Policy Gradient.**

## I. Introduction

In the presence of a rapidly evolving computer technology, data-driven models such as deep neural networks have seen large successes in most notably, fields such as computer vision, language models, and control systems. The ability to model systems without deterministically describing expected outcomes is of a large advantage concerning complex and nonlinear systems with significant noise and disturbances. Deep reinforcement learning, which utilizes a data-driven approach via the unsupervised learning of an extensive neural network, falls under this category.

Deep reinforcement learning determines the neural network parameters via trial and error, by iteratively interacting the network with its environment, and judging the outcome. By this method, the data used to train the network is not labelled, i.e. the exact output need not be known – only the extent to which it is desirable. This behaviour makes deep reinforcement learning an ideal approach for complex control system controllers, as one does not need to know the exact output over time required to achieve one's objective (for example, minimizing error). Instead, one can design the controller by using the objective itself as the model's desirability function (commonly referred to as the reward function).

Perhaps the most difficult aspect of implementing reinforcement learning is the way in which the network's parameters are update upon each iteration, for a given output and reward. Many algorithms have been developed to tackle this problem, including most notably the Twin-delayed deep deterministic policy gradient algorithm (TD3) [1], Proximal Policy Optimization (PPO) [2], and Q-learning [3]. In 2023, J. Cia et al developed a robust attitude controller using deep reinforcement learning and the TD3 algorithm. The paper reports successful simulation of the model for a variety of initial conditions, and in the presence of degrading actuator conditions. As an area of deeper investigation, Sections IV and V of this paper will describe the design and simulation of a simplified replication of the model derived by J. Cia et al. The immediately following Sections II and III offer relevant information to the controller replication, on neural networks, deep reinforcement learning, and the TD3 algorithm.

## II. Background Information

This section will offer background information targeted towards readers not familiar with reinforcement learning, neural networks, and the Twin Deterministic Policy Gradient (TD3). Its purpose is to serve as reference for both the literature review in Section III, and the following design and discussion involving a reinforcement learning controller created using the TD3 algorithm, in Sections IV and V.

### *A. Neural Networks*

The foundational work for neural networks was published in 1943 by McCulloch and Pitts, in which they described a simplified model of the workings of the human brain, and its ability to compute any arithmetic or logical function [4]. In recent years, this model has seen large amounts of success, primarily in part due to the increase in power, storage capabilities, and affordability of modern computers. The reason for this connection is due to the large reliance of neural networks on large data sets, and large parameter sizes of the model.

Neural networks are at their core a well-defined mathematical model that is capable of representing both complex and non-linear systems. The mathematical model is defined as such: consider the neural network shown in Fig. 1 below. This system takes in four inputs, and determines four outputs. The network consists of four layers total, two hidden layers each with six nodes, an input layer of four nodes, and an output layer of two nodes. The value of each node (excluding the input node) is the weight sum of each of the previous nodes, with an applied activation function. Each line connected to a node represents one previous node's weight contribution. For example, take any of the nodes in the third layer. The node's value is described as

$$f_a(w_1u_1 + w_2u_2 + w_3u_3 + w_4u_4 + w_5u_5 + w_6u_6 + w_7u_7 + w_8u_8),$$

where $f_a$ is the activation function, $w_i u_i$ represents the weighted contribution of each of the eight preceding nodes. All nodes within the network are computed in a similar manner. The parameters of the network to be determined using external data are each of the weights, $w_i$. In the considered example network, the total number of learnable parameters is given by

$$n_1 n_0 + n_2 n_1 + n_3 n_4 = (8)(4) + (8)(8) + (8)(2) = 104,$$

where $n_i$ represents the number of nodes in the $ith$ layer.

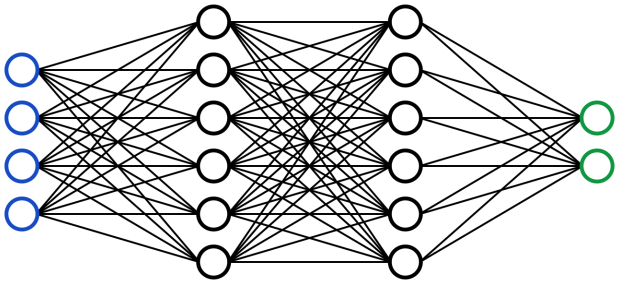


Fig. 1. A four-layer, 18 node neural network.

### B. Deep Reinforcement Learning

Reinforcement learning is a type of data-based training where the model being trained is a neural network. Unlike more commonly used methods of supervised learning where the data sets contain both a known input and known output, the "data" sets of reinforcement learning rely solely on the environment-determined results of an input, and the researcher's ability to give the result a score on how good or bad it is. Deep reinforcement learning refers to the large or "deep" number of layers and trainable parameters within the neural network. The term gained significant popularity when neural networks with large amounts of trainable parameters via a large number of hidden layers saw great success in fields of computer vision, and reinforcement learning.

The basic structure of a reinforcement learning model can be described using a step function, a reward function, an update function, and a neural network.

The neural network, or **policy network**, is the model whose system will be the result of training. The goal of reinforcement learning is to iterate through an environment, and update this network's parameters, in the hopes of finding a system that accurately defines the relationship between its input and output. The input to the neural network is defined as the **state** of the system, and its output is defined as the **action.**

The step function consists of calculating the next state, $n + 1$, for a given action and current state, $n$. This function defines how the environment will react to receiving some action.

The reward function defines what result of the model is good or bad upon each iteration. The training algorithm implemented will be designed to maximize the reward received by the algorithm.

The update function consists of the methodology employed to update the weights or parameters of the neural network upon each iteration. Typical parameter-dimensional spaces defined by the neural network structure are too large to explore in whole. A method is required to explore the space efficiently and update the parameters so that they are headed in a direction that maximizes the reward function. Examples of algorithms are the Q-network algorithm, and TD3 algorithm.

### C. Twin Deterministic Policy Gradient

The Twin Deterministic Policy Gradient or TD3 algorithm was introduced by in 2018 by Scott Fujimoto et al. as an improvement upon the existing Deep Deterministic Policy Gradient (DDPG) [5] for a continuous action space. Excluding the policy network, the TD3 utilizes two additional neural networks known as the critic networks. These networks evaluate the action-reward function, a measure of the expected return of the reward for a given action. The minimum of the two action-rewards or Q-values are then taken. By taking the Q-value as the minimum of two networks, the TD3 algorithm solves many of the instabilities of just one actor network found in the DDPG algorithm.

## III. Literature Review

### A. Introduction

This section will serve as a literature review on reinforcement learning in control systems. Its purpose is to provide a larger scope of the material related to the design and simulation following in later sections. The 1st paper reviewed, by J. Cia et al, is the paper for which the mentioned simulation and design is based on. All papers have been published by the Institute of Electrical and Electronics Engineers (IEEE).

As the focus of the investigation is regarding reinforcement learning, special attention is paid towards specific hyperparameters regarding the reinforcement learning training. Since different algorithms will be used for network updates, the most notable parameters considered are the **episode count, actor learning rates,** and of course the **algorithm** used. The motive for recording such information is to give context to the altitude controller reconstruction and following results in Section IV and Section V, respectively.

### B. Robust Attitude Controller Designation of Launch Vehicle under Actuator Failure Condition via Deep Reinforcement Learning Algorithm (2023)

Launch Vehicle Attitude controllers serve the purpose of controlling the orientation of launch vehicles during flight. This is essential, as launch vehicles travel in the direction they are orientated in. At the 35th Chinese Control and Decision Conference (CCDC) in 2023, J. Cia et al presented their work on the successful design and simulation of a robust attitude controller designed with deep reinforcement learning and the TD3 algorithm [6]. The simulation is completed with and without a decaying actuator failure (reduced model-output effectiveness over time) showing remarkably low errors, and stable convergence of the trained model.

The group was able to reach their proposed model of training for 200 episodes, a learning rate of 0.0001, utilizing the TD3 algorithm.

### C. Research on Cooperative Control of Traffic Signals based on Deep Reinforcement Learning (2023)

L. Fann et al. purpose both an index, and algorithm aimed at solving the problem of regional traffic congestion, using deep reinforcement learning via a Deep Q-Network (DQN) algorithm [7]. The group simulates the algorithm using Simulation of Urban MObility (SUMO), an open-source simulation tool for urban area traffic, and compare the results against three other known algorithms. They conclude that their proposed deep reinforcement learning algorithm obtains better performance in average wait time, and average time loss.

The group reached their proposed model over 100 episodes, a 0.01 learning rate, and utilizing the DQN algorithm.

### D. Passivity-Based Online Reinforcement Learning for Real Time Model-Free Overhead Crane System Control (2022)

H. Zhang et al. introduce a novel model-free, online reinforcement learning model to control the position payload position of over-head cranes [8]. The proposed model is called the Passivity-Based Online Deterministic Policy Gradient (PB-ODPG) model. Unlike previously examined papers, the algorithm and thus the way in which the network parameters are updated here is novel. The structure designed by this group is based on an actor-critic architecture as we have seen before in both the DDPG and TD3 algorithm, though the PB-ODPG model has a strong design on focus on **online learning** (the controller network itself is updating in real-time as it is being implemented). Previous networks in Subsections A. B., and C. all utilized offline training, as the controller's neural network was trained in a simulated environment before using the controller in a simulation.

The number of episodes applied to this model is nonapplicable, as the model is online and continuously training during implementation. The algorithm used to update weights is the novel model proposed in this model known as the PB-ODPG model. The actor learning rate is not specified in the paper.

### E. Deep Reinforcement Learning for Control Design of Quantum Gates (2022)

In the field of quantum computing, there exists many counterparts to that of classical computing, for instance quantum bits vs binary bits, and quantum gates vs logical gates. The key difference is that quantum bits represent a quantity adhering to the fundamental principles of quantum mechanics, such as the intrinsically statistical nature when measuring quantum information. In this paper, S. Hu, C. Chen, and D. Dong develop multiple reinforcement learning controllers aimed at the control of quantum gates using the TD3 algorithm [9]. The group simulates the found controllers on four different kinds of gates including three one-qubit gates and one two-qubit gate. The group finds better performance against known control algorithms in the Gradient Ascent Pulse Engineering (GRAPE) algorithm [10] and the Differential Evolution (DE) algorithm [11].

The group reports various episode lengths ranging between 100-400 for the different trained models corresponding to different gate control signals. The algorithm was TD3, as was used by the group discussed in Subsection A. No learning rate is stated.

### F. Robotic Control Mechanism Based on Deep Reinforcement Learning (2023)

Z. Pan et al. propose a novel deep reinforcement learning architecture in the control of robotic mechanisms [12]. The group simulated their own model against more well-known algorithms in Q-learning and Proximal Policy Optimization (PPO). They find their own model outperforms both competing algorithms in time delay, and accuracy for a range of robotic signals to control. The novel algorithm proposed by the group seems to have a very basic reinforcement learning architecture (as seen in the earliest developments of the area) for online algorithms. However, it very notably contains a function to update the reward function itself during the online implementation of the algorithm. In contrast, the majority of algorithms do not change the function itself, but rather use the well-defined reward function as a key component of the iterative process in training.

The algorithm used by Z. Pan et al. is novel and without name, and the learning rate is not mentioned. The concept of episodes does not apply, as the group indicates the developed model is trained during implementation (online).

### G. Discussion

The preceding Subsections B-F investigated the findings of five separate papers on the topic of reinforcement learning in control systems. Each paper proposed a new controller related model or controller itself. The models were trained from a variety of algorithms including TD3 and DQN algorithm, as well as two groups who proposed novel algorithms that they themselves developed. These algorithms proposed new strategies specific to the targeted problem, while building upon the architecture and knowledge of previous reinforcement learning algorithms. For example, as one might expect, both novel algorithms utilized the gradient descent method, a method of updating the neural network parameter core to many topics regarding the field of neural networks.

The topics to which reinforcement learning was applied varied greatly between the five papers. These included control in quantum gates, robotics, launch vehicles, traffic congestion, and over-head cranes. All five papers found success in the respective problems via the implementation of reinforcement learning.

## IV. Design

This section will present a design for a one-dimensional attitude controller of a launch vehicle using reinforcement learning and the TD3 algorithm. The design, and its simulation in the following Section V are a simplified replication of the robust attitude controller found by J. Cia et al., discussed in Subsection B of the literature review conducted in Section III. The one-dimensional controller designed here is created and simulated using MATLAB's reinforcement learning toolbox.

The design is split into three subsections: a problem statement including a block diagram, the reinforcement learning environment, and model training.

### A. Problem Statement

Consider the closed-loop block diagram shown below in Fig. 2. The goal is to obtain a neural network such that it will act as an attitude controller. This controller at any time $t$ will take in the orientation and angular velocity of the launch vehicle, and output the required torque to reach/maintain the desired angle. The desired angle will be set to 0 degrees with respect to a fixed reference frame. To simulate instabilities during flight, the initial conditions of the controller will be set to a random initial angle between -20 and 20 degrees. The controller will be trained and simulated in its effectiveness to respond to the random offset.

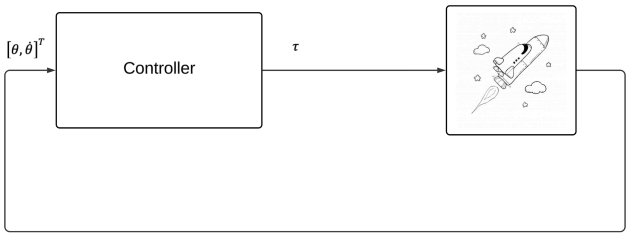



Fig. 2. Block diagram of one-dimensional attitude controller design

In one dimension the equations of motion describing a launch vehicle can be described simplistically with a 2nd order derivative in the form of

$$\tau = I\ddot{\theta},$$

where $\tau$ is the applied torque, $I$ is the moment of inertia of the launch vehicle about its centre of mass, and $\theta$ is the orientation of the launch vehicle with respect to a fixed reference frame.

### B. Environment

To create the reinforcement learning model we define the environment, and the hyperparameters of the TD3 algorithm to be implemented. The environment consists of determining the next state given a certain action (step function), as well as the reward function for the new state as a result of said action. The weights of the neural network are then updated via the TD3 algorithm. This process continues to the next time sample of $t + T_s$, and repeats. This architecture is shown via a block diagram below in Fig. 3.

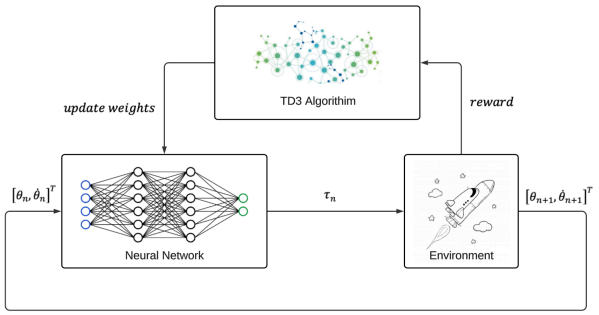



Fig. 3. Block diagram of one-dimensional reinforcement learning architecture

The step function calculates the next state by finding $\theta_{n+1}$ for a given state $\theta_n$ and action $\tau_n$. To calculate $\theta_{n+1}$, we take the input of the neural network to be a two-dimensional continuous input space, defined by $\theta$ and $\dot{\theta}$, the orientation and angular velocity in radians per second, respectively. For a sample time of $T_s$, the current state to next state step function is given as

$$\left[\theta_{n+1}, \dot{\theta}_{n+1}\right]^T = \left[\theta_n, \dot{\theta}_n\right]^T + T_s \left[\dot{\theta}_n, \frac{\tau}{I}\right]^T.$$

The reward is taken to be the negative of the absolute steady-state error between the next state orientation, $\theta_{n+1}$, and the desired angle, 0 degrees. This is given as

$$reward = -abs(\theta_{n+1})$$

To create the TD3 algorithm, MATLAB's reinforcement learning toolbox is used. Based off the literature review and paper conducted by J. Cia et al, the following list of hyper-parameters was used to define the TD3 algorithm used.

TABLE I
LIST OF HYPER-PARAMETERS

| Hyper-parameters | Definition | Values |
|---|---|---|
| $\boldsymbol{n}^{(Actor)}$ | Hidden layers and their number of nodes of the actor network | [64, 64, 64, 64] |
| $\boldsymbol{n}^{(critic)}$ | Hidden layers and their number of nodes of the critic network | [64, 128, 128, 64, 64] |
| $[\alpha_{actor}, \alpha_{critic}]$ | | [1e-4, 1e-4] |
| $T_s$ | The time duration of each time step, i.e. the sample time | 0.1 seconds |
| $[\Delta_{actor}, \Delta_{critic}]$ | Gradient threshold | [1, 1] |
| $f_T$ | Target update frequency | 2 |
| $f_P$ | Policy update frequency | 2 |
| $\tau$ | Target smooth factor | 0.005 |
| $\gamma$ | Discount factor | 0.99 |
| $\mu_N$ | Mean exploration noise | 0 |
| $\mu_S$ | Mean target policy smoothing noise | 0 |
| $\sigma_N$ | Standard deviation of exploration noise | 0.2 |
| $\sigma_S$ | Standard deviation of target policy smoothing noise | 0.2 |

### C. Model Training

The model defined in Subsection B. is trained within MATLAB over 900 episodes, and max episode time of 10 seconds, or 1000 steps. The results are shown below in Fig. 4, where the episode reward is defined as the summation of all rewards within the episode, and the average reward is

calculated via the 20 most recent reward values. After 900 episodes, the model converges around an average episode reward of -200.

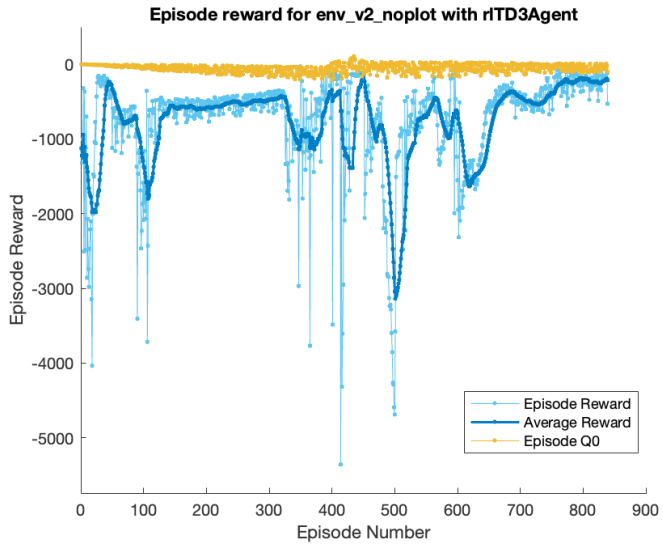



Fig. 4. Episodic reward during the training process

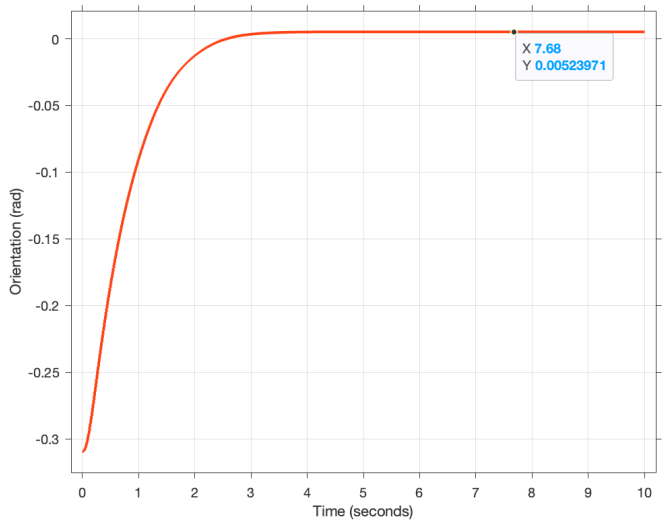
## V. Results

To obtain the controller as described in Subsection A of Section IV, the actor network is taken from the reinforcement learning model and simulated using the already defined environment of the launch vehicle within MATLAB. Fig. 5 shows the results of an example of such a simulation below, where the initial condition is randomly set -0.31 rad or -17.8 degrees, and angular velocity of 0 degrees per second.

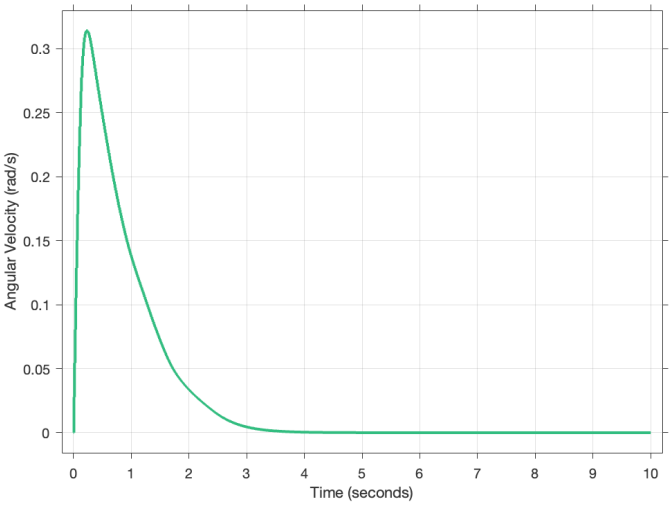
For this example, we find that over a period of 3 seconds, the orientation of the launch vehicle converges to a stable value of 0.005 rad, or 0.28 degrees. The steady state error is then 0.28 degrees. The angular velocity and applied torque both converge to 0 in units rad/s and Nm respectively, as expected from the stability of the orientation of the launch vehicle.

To generate an appropriate analysis of the controller, 100 simulations were conducted on the controller, finding average steady state errors within +/- 0.3 degrees, and stable convergence.

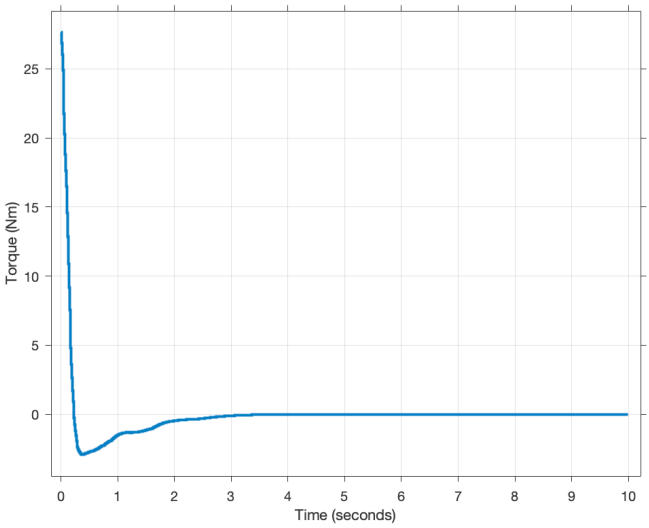
(a)




(b)




(c)



(d)

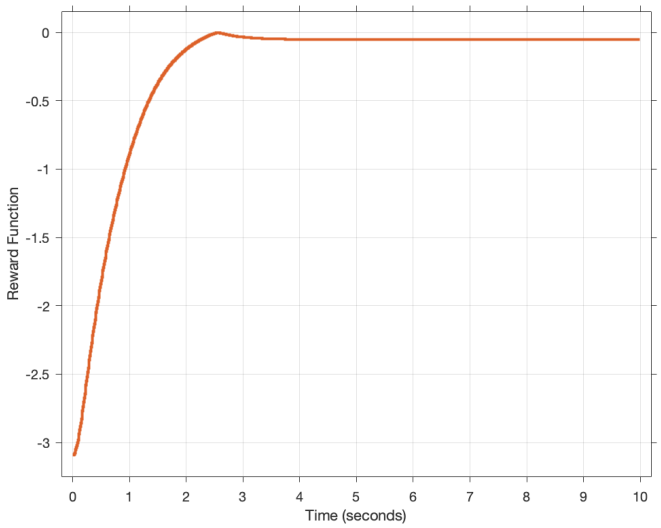



Fig. 5. Simulation result of trained attitude controller

## VI. Discussion

The simulation in Section V found the controller generated a stable response with a steady state error of +/- 0.3 degrees with respect to the centre of mass. For large launch vehicles, wight heights around, for example, 50m, this corresponds to a straight-line error between desired and actual displacement of the rocket tip of 13 cm – a substantial error. In comparison, J. Cia et al. present a 0 steady-state error controller. While the form of the controller and its stability are desirable, the steady-state error leaves future improvements to be made.

J. Cia et al. report a training episode length of only 200 episodes, where the model appears to converge around episode 20-30. This is in large contrast to the 900-episode convergence of the model found here, despite being simplified. While the model architecture and many of the key hyper-parameters were replicated between papers in the TD3 algorithm, many assumptions were made including the environment, step function, and a list of other details of the TD3 algorithm not specified by the group.

While many configurations of the training model were attempted including various reward functions (including time dependencies), different actor and critic networks (including the number of hidden layers/nodes), and TD3 parameters, the majority of training resulted in unstable responses or divergences in the Q0 value during training.

## VII. Conclusion

Five papers published within the last two years on control systems via reinforcement learning and various algorithms were investigated. The proposed model from one of these papers on attitude control was investigated further, via an attempt to replicate the findings. The zero steady-state error and short training episodes found by J. Cia et al. were not able to be replicated. Instead, a low steady-state error model was found to control the attitude of a launch vehicle in one dimension, to moderate success. Future work in improving the model found in this paper include extending the simulation time, and careful addition of time dependent factors into the reward function in hopes of reducing steady state error.

## References


[1] Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *In International Conference on Machine Learning (ICML)*.

[2] Watkins, C.J.C.H., & Dayan, P. (1992). Q-learning, *Machine Learning,* 8(3-4), 279-292.

[3] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms.

[4] McCulloch, W.S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5*, 115–133 (1943).

[5] T. P. Lillicrap, J. J. Hunt and A. Pritzel, "Continuous Control with Deep Reinforcement Learning" 2015.

[6] C. Jia, X. Liu, Z. Wang, Q. Gong and X. Huang, "Robust Attitude Controller Designation of Launch Vehicle under Actuator Failure Condition via Deep Reinforcement Learning Algorithm," *2023 35th Chinese Control and Decision Conference (CCDC)*, Yichang, China, 2023, pp. 3223-3228.

[7] L. Fan, Y. Yang, H. Ji and S. Xiong, "Research on Cooperative Control of Traffic Signals based on Deep Reinforcement Learning," *2023 IEEE 12th Data Driven Control and Learning Systems Conference (DDCLS)*, Xiangtan, China, 2023, pp. 1608-1612.

[8] H. Zhang, C. Zhao and J. Ding, "Passivity-Based Online Reinforcement Learning for Real Time Model-Free Overhead Crane System Control," *2022 34th Chinese Control and Decision Conference (CCDC),* Hefei, China, 2022, pp. 4116-4121.

[9] S. Hu, C. Chen and D. Dong, "Deep Reinforcement Learning for Control Design of Quantum Gates," *2022 13th Asian Control Conference (ASCC)*, Jeju, Korea, Republic of, 2022, pp. 2367-2372.

[10] Khaneja, N., Reiss, T., Kehlet, C., Schulte-Herbrüggen, T., & Glaser, S. J. (2005). Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms. Journal of Magnetic Resonance, 172(2), 296-305.

[11] Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11(4), 341-359.

[12] Z. Pan, J. Zhou, Q. Fan, Z. Feng, X. Gao and M. Su, "Robotic Control Mechanism Based on Deep Reinforcement Learning," *2023 2nd International Symposium on Control Engineering and Robotics (ISCER)*, Hangzhou, China, 2023, pp. 70-74.